Fabrication-Aware Design with Performative Criteria

THÈSE N^{O} 6429 (2015)

PRÉSENTÉE LE 6 FÉVRIER 2015

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE D'INFORMATIQUE GRAPHIQUE ET GÉOMÉTRIQUE
PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Yuliy SCHWARTZBURG

acceptée sur proposition du jury:

Prof. E. Telatar, président du jury
Prof. M. Pauly, directeur de thèse
Prof. M. Alexa, rapporteur
Prof. R. Hersch, rapporteur
Prof. T. Weyrich, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2014

See first that the design is wise and just: that ascertained, pursue it resolutely; do not for one repulse forego the purpose that you resolved to effect.
— Aesop

Wings are a constraint that makes it possible to fly.
— Robert Bringhurst

To my parents, my brother, my extended family and friends, and most of all my girlfriend—everyone in Switzerland, the United States, and around the world—that has supported me during these years.

Acknowledgements

Scientific work is not possible these days in solitude, and there have been so many people that have been important in large ways and small in making it possible for me to write this thesis. What follows is not in any particular order and is by no means exhaustive, so I profusely apologize if I have left out anyone.

Most of all, I thank my advisor Mark Pauly who has been great since the start. I'm immensely thankful that he took me in at the start when my options had run dry with my previous lab. He took a bet on me and I hope it has paid off. He has been supportive, insightful, and inspiring, and I know that our future collaborations will be as great as the preceding ones.

I want to thank my co-authors immensely as much of the work in this thesis is due in large part to them. Their names are listed in Section 1.3. I also thank my thesis committee for all their help—Roger Hersch, Tim Weyrich, Marc Alexa, and Emre Telatar. Thank you for all your helpful comments and questions.

I would like to thank my mother and father for making sure I was always putting my dissertation above all other concerns except for, of course, my health. They have constantly supported me in everything I've done, and have been very supportive in me being so far away from home. They have always been amazing parents and I greatly appreciate everything they have done for me to get me to this point. I thank my brother as well, especially for keeping me updated on everything that has been happening while I'm away from the United States. I am very happy to have gone through my PhD at the same time he was going through his educational and professional journey to become a lawyer.

I should of course thank my girlfriend, Lyvia Fishman, who went through the ups and downs with me the whole way. I thank her parents, Cathy and Bernard, who took me in and provided me a second home in Switzerland. And her brother Ron, who as well helped me in translating the abstract. I also thank the entire Fishman extended family–grandparents, aunts, uncles, cousins–and Lyvia's friends who have welcomed me and have been great friends for the last 4-5 years. I don't know where I would be without them all.

I cannot forget all my friends in the United States, but there are (fortunately) too many to list. I want to particularly acknowledge Laura Berger, Michael Yagliyan, Jared Harwayne-Gidansky, Eli Halpern, David Tuchman, Elisa Mala, Javier Rodriguez, Kevin

Teoh, and Sara Foley, but there are so many people not in this list who are just as important to me.

My work has benefited enormously from long and fruitful discussion and collaboration with the members of my lab. On the architectural geometry side—from the beginning of my time with the lab, Sofien Bouaziz and Mario Deuss have provided me with not just great scientific discussion and many ideas but have also been great friends throughout it all. It is rare to find people who are not just greatly skilled in research but can also relax and have fun rather often, making sure to always strike a good balance between the two. I want to thank Bailin Deng for discussion, help, and for putting up with me as a roommate many times at conferences. I've had the pleasure of having Romain Testuz as a student, working with him on several projects as mentor, and now working alongside with him. I cannot wait to see where we will go in our collaboration. Andrea Tagliasacchi and Alexandru Ichim are recent additions but I feel like they have been here the whole time–they have provided me with much help and also nicely rounded out our group of friends. Duygu Ceylan, Minh Ngoc Deng, Stefan Lienhard and Boris Neubert, while being far away physically at the end of the hall, have in no less way contributed to lots of great conversation and in making my experience great. There are also those who left early in my time here but have nevertheless made a mark—Thibaut Weise, Michael Eigensatz, and Hao Li. Nor can I forget those that are not members of LGG but nevertheless those currently or formally at EPFL—Cheryl Lau, Horesh Ben Shitrit (as well his wife Adar), Heather Miller, Arash Farhang, Mike and Alisa Ferdman, Petr Susil, Seth Flaxman, Chris Evans, Evan Williams, Albrecht Lindner, Radhakrishna Achanta, Trevor Patt, Nathaniel Zuelke, among many others. A special thanks as well to our wonderful secretary Madeleine Robert, who has helped me out countless times.

There are also those in the extended community who I have to thank for much discussion and being friends in the field away from home. This non-exhaustive list (as I'm probably forgetting a great many people) includes Keenan Crane, Justin Solomon, Fernando de Goes, Daniele Panozzo, Chengcheng Tang, Yonghao Yue, Brian Amberg, Alec Jacobson, Etienne Vouga, Breannan Smith, Christopher Batty, and so many more.

Outside of Computer Graphics, I have spent a lot of time with the Catalyst Theater Group, doing improvisational comedy and acting in theatrical productions. It is a different type of scientific collaboration, but just as important to me. Thank you particularly my co-stars and friends Adria Le Boeuf, Kyle Gustafson, Luis Miguel Fidalgo, Friederike Bienert, Giulia Beanato, Isabel Schmid, Katie Ridout, Shahin Tavakoli, Luca Bartesaghi, Diego Cortez, Paula González-Rubio, Diego Gonzalez, Gustavo Ruiz, Charles Mullon, Derek Setter, and all the new people. I am very excited to be working with the Catalyst again for The Blue Butterfly. In the same vain, I have also acted with students of the UNIL English department for two years. I want to thank my co-stars from the Sweet Sorrow Theater Group, including particularly Florence Rivero, Raphaël Meyer,

Sami Veillard, and Fredrik Blanc. And also all my co-stars in At First Sight, particularly Cathy Rime, Max Borg, and Gavon Balharry.
Throughout my thesis, I have received a lot of help in areas outside of my immediate expertise. For fabrication help with laser cutting and milling the planar pieces, I thank Stephane Grandgirard, Russell Loveridge, Christopher Robeller, Hans Ulrich Buri, and the IBOIS lab.
For the caustic design project, I would like to thank Alfred Thomas, Nicolas Favre-Victoire and Claude Cheseaux from the EPFL ATPR workshop and Florian Rist of TU Vienna for a lot of help and patience with fabrication. I also thank Tuan Anh Le for contributing parts of the code, Sofien Bouaziz for valuable discussion, and Yonghao Yue. Finally, thanks to the Philippe Halsman Archive for providing the Einstein photo.
Lastly, I want to acknowledge EPFL and the funding agencies that have supported me financially. The last year of my thesis research has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007–2013) / ERC Grant Agreement n. 257453; COSYM.
Thank you to everyone including those I forgot to list, I cannot have done this without your help, encouragement, support, and love.

Lausanne, 3 October 2014 Y. S.

Abstract

Artists and architects often need to handle multiple constraints during design of physical constructions. We define a *performative constraint* as any constraint on design that is tied to the performance of the model–either during fabrication, construction, daily use, or destruction. Even for small to medium scale models, there are functional criteria such as the ease of fabrication and the assembly process, or even the interplay of light with the material. Computational tools can greatly aid in this process, assisting with the lower-level performative constraints, while the designer handles the high-level artistic decisions. Additionally, using new fabrication methods, our tools can aid in lowering the difficulty of building complex constructions, making them accessible to hobbyists. In this thesis, we present three computational methods for designing with different approaches, each with a different material, fabrication method, and use case.

The first method is a construction with intersecting planar pieces that can be laser cut or milled. These 3D forms are assembled by sliding pieces into each other along straight slits, and do not require other support such as glue or screws. We present a mathematical abstraction that formalizes the constraints between pieces as a graph, including fabrication and assembly constraints, and ensure global rigidity of the sculpture. We also propose an optimization algorithm to guide the user using automatic constraint satisfaction based on analysis of the constraint relation graph. We demonstrate our approach by creating several small- to medium-scale examples including functional furniture. We also extend our approach based on recent work and demonstrate other performative constraints such as a target silhouette.

The second method presents a solution to a problem proposed by the LEGO® group in 1998, building a 3D sculpture out of existing building blocks that can be found in many homes. Starting from the voxelization of a 3D mesh we merge voxels to form larger bricks, and then analyze and repair structural problems based on a graph representation of the block connections. We then output layer-by-layer building instructions to allow a user to quickly and easily build the model. We also present extensions such as hollowing the models to use less bricks, limiting the number of bricks of each size, and including color constraints. We present both real and virtual brick constructions and associated timings, showing improvements over previous work.

The final case presented tackles the inverse design problem of finding a surface to produce a target caustic on a receiver plane when light is refracted or reflected. This is

an example where the performative constraint is the principal driver of the design. We introduce an optimal transport formulation to find a correspondence between the incoming light and the output target light distribution. We then show a 3D optimization that finds the surface that transports light based on the correspondence map. Our approach supports *piecewise smooth* surfaces that are as smooth as possible but allow for creases, to greatly reduce the amount of artifacts while allowing light to be completely diverted producing completely black regions. We show how this leads to a very large space of high-contrast, high-resolution caustic images, including point and line singularities of infinite light density as well as photo-realistic images. Our approach leads to surfaces that can be milled using standard CNC milling. We demonstrate the approach showing both simulated and fabricated examples.

Key words: architectural geometry, digital fabrication, fabrication-aware design, computational design, 3D optimization, inverse surface design, complex assembly, caustics, lego

Résumé

Les artistes et les architectes doivent gérer plusieurs contraintes lors du design de constructions physiques. Les contraintes dite *performatives* sont liées à la performance du modèle dans sa fabrication, sa construction, son utilisation quotidienne ou sa destruction. Même pour les modèles de petite et moyenne taille, il existe des critères fonctionnels tels que la facilité de fabrication et d'assemblage ou l'interaction de la lumière avec la matière. Les outils informatiques aident grandement dans ce processus : ils facilitent la prise en charge des contraintes performatives et permettent ainsi aux designers de se focaliser sur les décisions artistiques. Grâce à de nouvelles méthodes de fabrication, nos méthodes facilitent la construction d'ouvrages complexes et les rendent ainsi accessibles aux amateurs. Dans cette thèse nous présentons trois approches avec différentes méthodes de fabrication.

La première méthode permet de construire des formes 3D en assemblant des pièces planes découpées au laser (ou fraisées). Les pièces sont fixées l'une à l'autre en glissant dans des fentes prévus à cet effet, ni vis ni colle n'est donc nécessaire. Nous présentons une abstraction mathématique qui formalise, sous forme de graphe, les contraintes entre pièces ; notamment les contraintes d'assemblage et de fabrication, qui assurent la rigidité globale de la sculpture. Nous proposons également un algorithme d'optimisation en utilisant la satisfaction automatique des contraintes basée sur l'analyse du graphe de relation des contraintes. Nous démontrons notre méthode avec plusieurs exemples de petite et moyenne taille, dont des meubles. Nous expliquons également notre approche en nous basant sur des travaux récents et montrons d'autres contraintes performatives tel qu'une silhouette cible.

La deuxième méthode offre une solution à un problème posé par le groupe LEGO en 1998 : construire une sculpture 3D avec les briques de la marque. En partant de la voxelisation d'un maillage 3D, nous fusionnons des voxels pour former de plus grandes briques, ensuite nous analysons et réparons les problèmes structurels en se basant sur un graphe représentant les connexions entre bloques. Il en résulte des instructions de construction couche-par-couche permettant à un utilisateur de créer facilement et rapidement le modèle. Nous présentons également des extensions, tels que des modèles creux nécessitant moins de briques, une limite du nombre de briques de chaque taille, et des contraintes de couleur. Nous présentons des exemples réelles et virtuelles, leur temps de calcul, ainsi que les améliorations apportés aux précédents travaux.

La dernière méthode s'attaque à un problème de conception inverse qui consiste à créer une surface produisant une caustique cible sur un plan lorsque la lumière est réfractée ou réfléchie. C'est un exemple où la contrainte performative est le principal moteur de la construction. Dans un premier temps, nous introduisons une formulation de transport optimal afin de trouver une correspondance entre la lumière entrante et la distribution de la lumière sortante. Nous présentons ensuite une optimisation 3D qui permet de trouver la surface altérant la lumière afin de reproduire la carte de correspondance. Notre méthode reproduit des surfaces aussi lisses que possible mais permet les plis, de bien réduire la quantité d'artefact en divergeant complètement la lumière, produisant ainsi une région sombre. Nous montrons comment cela produit des contraste élevé, des images caustiques haute résolution – notamment des points et des lignes de densité lumineuse infinie – ainsi que des images photo-réaliste. Le produit fini consiste en une surface altérée avec un fraisage standard CNC. Nous montrons la réussite de la méthode avec des exemples simulés et réels.

Mots clefs : géométrie architecturale, fabrication numérique, conception orientée fabrication, conception par ordinateur, optimisation en 3D, conception inversée de surface, assemblage complexe, caustiques, lego

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Even artistic endeavors are subject to the laws of physics. Artists and architects have traditionally relied on training or intuition when designing and constructing projects. They are always constrained by the material and tools they use. The tools evolve, but artists, in the race to gain the attention of a public that craves novelty, constantly want to push past the limitations of the tools. When their own education and experience is not enough, they rely on outside help—technicians and engineers—and there is a constant negotiation between artistic intent and physical limits. This process can get prohibitively time-consuming and expensive.

As more work is being done digitally, we can harness the power of computation to give designers tools that can lessen the time spent and the workload of the negotiation process. We can give designers tools that respond to artistic intent and allow them to immediately see if their design is realizable. More than that, we can give them clues or push them into feasible configurations so that staying within physical limits is seamless. This way once the designer is ready to fabricate, they are sure that the design is already close to the final construction.

With computational tools, not only can existing processes be accelerated, but we can also enable algorithms to contribute to the design. As Axel Killian [Kil06] writes, "Constraints are usually perceived as limitations in design exploration. With computational models and the ability to model even complex dependencies their role may be changing towards that of design drivers." Killian introduces a new paradigm–*bidirectional modeling*–where the final product comes from both the designer and the constraints, which are enforced by software. The artistic effort and the constraint satisfaction happens in a cyclical fashion, iterating on the design, until it is completed.

We define a *performative constraint* or equivalently, a *performative criterion*, as any

constraint on design that is tied to the performance of the model in a certain physical context. This can be during the fabrication, the construction, in the years it is used while standing be it normal use or extreme cases, and even during and after destruction. In recent years, architects have called for *performative design*, or a process where the constraints are inherent to the design itself. Rivka Oxman [Oxm09] writes, in traditional architecture the term performance in design "is associated with the analytical act of a posteriori evaluation [...] Design generation is followed by performance evaluation, which may condition further processes of generative modification [...] The transition towards performative models attempts to prioritize performance and to create a seamless and integrated process of performance-based design."

An example of a performative constraint being used in architectural software today is evaluation of environmental criteria. A building (such as the Greater London Authority Headquarters building by Foster and Partners) must be optimized for natural light, ventilation, acoustical performance, and low energy consumption. The existing process is modifying the current models iteratively with feedback from structural and environmental engineers until each objective has been satisfied [Oxm09]. Of course, this gives an outcome that is far from optimal. Without computational tools, it is difficult to be concious of these criteria to the initial design, as the complexity and interdependence makes for an almost impossible task even for an experienced architect. Recently, we are seeing architects themselves developing systems to combine evaluation software with generative design and optimization [LNR10, SNR11, SY13]. However, the evaluation software is not built for easy integration with optimization systems and without the proper knowledge and careful modeling of constraints, these systems often suffer from inefficient global optimization and non-optimal results.

It is impossible to design with every constraint in mind a-priori and the research community is far from combining all these aspects into one tool. We take a first step in this direction by looking at a subset of performative criteria that apply to constructions of small- and medium-scale—such as stability of the model, ease of fabrication depending on the chosen fabrication technology, assembly given the particular physical criteria, or similarly to the environmental criteria, side effects of the construction such as interaction with light and color.

As technologies such as additive manufacturing and 3D printing become more common and accessible, there is also a need for tools that take advantage of those technologies and enable complex designs. The lower the barrier of entry, the more desire there is to differentiate and to push the limits of the machines. Improving the hardware is one option, but by modelling the geometry and developing algorithmic tools, we can look at the problem at a higher level and thereby enable more intricate and more surprising

constructions.

With a combination of computational models, algorithms, and *computer numerical control* (CNC) machining, we can even go beyond the imagination of an artist, and allow for never-before-seen fabricated results and effects. This thesis explores the aid of computation with fabrication for three design paradigms—building constructions out of intersecting planar pieces, building models out of LEGO® construction blocks, and making controlled caustic effects out of transparent 3D pieces. In each of these situations, due to the complexity of the fabrication tool, material, and other physical constraints, a computational method is needed. Our methods are able to greatly aid in design, and particularly in the final case, enable physical constructions that were never possible before.

1.1 Overview

The primary motivation of this work is enabling physical structures using computational tools. These tools are needed to build designs that are fabricable, assemblable, and fulfill other performative constraints as needed by the project. Within this space there are several challenges, and each chapter in this thesis addresses an example.

Designing 3D models to be fabricated is a very different task than designing models to be presented in film or video games. Traditionally, computer graphics has focused on the latter industries, but since computer graphics first became possible, there has been demand to use similar tools in other industries such as architecture and mechanical engineering. To deal with this demand, there have been many commercial projects such as AutoCAD® and Solidworks®. While these are great at communicating designs, they still allow a great many degrees of freedom to the designer that will let the designer stray from physically realizable designs. Tools have been developed to let users test the feasibility of designs, but a common problem is they will not propose solutions. Or if they do, these solutions are often contradictory to the artistic intent of the original design before testing. Therefore, tools must be developed that keep the project feasible throughout. Chapter 3 presents a specific instance of this problem where an architect or artist wants to fabricate a model easily and cheaply out of intersecting planar pieces that can be cut from a laser cutter. The artist is constrained to stay within a space of feasible, assemblable designs. This space is defined through simple geometric constraints on such that an optimization can be designed around it. This chapter provides an example of using such constraints along with graph theory to build a system where designer input and optimization work hand-in-hand, in which the designer inputs high level decisions

while the system handles the low-level constraints.

However, suppose a user does not have access to a CNC machine. In this case, he or she can still prototype a 3D object out of materials they already have, quickly and cheaply. Chapter 4 proposes a system which uses LEGO® bricks as an example. These toys are often found around the home and by using our system, they can be repurposed to construct 3D models. In this chapter, the system works as a post-rationalization of an existing design but the constraints are still based on geometry and assembly. This enforces the idea of using graph theory to model assembly and using optimization to handle physical constraints.

Finally, computational tools can help designers and fabricators use CNC machines to their maximum potential. Using an input such as an image, we can enable designers to use physical constraints to their advantage. In Chapter 5, we propose a method to construct pieces of transparent or reflective material that are able to redirect light to "paint" a target image with light. This chapter provides an optimization that handles the physical constraints of the CNC system being used (in our case 3/5-axis milling), but goes further in also optimizing for the performative constraint of a target irradiance produced by the light redirection due to the piece geometry.

1.2 Contributions

The core contributions of this thesis are:

- The definition of a constraint space of 3D objects composed of planar intersecting pieces that is derived from requirements for fabrication, stability, and assembly,
  - an optimization algorithm for finding feasible configurations that satisfy all constraints, and
  - an interactive system that enables effective design exploration within the space of feasible solutions,
  - as well as the ability to add performative criteria such as silhouette constraints.
- The definition of a graph-based model of stability for LEGO bricks that is shown to outperform more complicated heuristics,

  - an optimization algorithm for quickly solving for this system,
  - and the ability to use other performative constraints, such as color, piece limits, and variable solidity.
- A new optimization algorithm for inverse caustic design based on optimal transport,
  - resulting in a continuous, piecewise smooth surface reducing artifacts,
  - enabling high-contrast target images including completely black areas,
  - point and curve singularities of infinite light density in the target image,
  - and non-uniform input light distributions with free boundaries.

1.3 Publications

This thesis covers the following peer-reviewed publications:

- Yuliy Schwartzburg and Mark Pauly. Design and optimization of orthogonally intersecting planar surfaces. In *Computational Design Modelling*, pages 191–199. Springer, 2012
- Yuliy Schwartzburg and Mark Pauly. Fabrication-aware design with intersecting planar pieces. *Computer Graphics Forum (Proc. of Eurographics 2013)*, 32(2pt3):317–326, 2013
- Romain Testuz, Yuliy Schwartzburg, and Mark Pauly. Automatic generation of constructable brick sculptures. In *Eurographics 2013-Short Papers*, pages 81–84. The Eurographics Association, 2013
- Yuliy Schwartzburg, Romain Testuz, Andrea Tagliasacchi, and Mark Pauly. High-contrast computational caustic design. *ACM Trans. Graph. (Proc. of SIGGRAPH 2014)*, 33(4):74:1–74:11, July 2014

In addition, the following publications were published during the same time period but are not explicitly addressed here:

- Sofien Bouaziz, Mario Deuss, Yuliy Schwartzburg, Thibaut Weise, and Mark Pauly. Shape-up: Shaping discrete geometry with projections. *Computer Graphics Forum*, 31(5):1657–1667, 2012

- Bailin Deng, Sofien Bouaziz, Mario Deuss, Juyong Zhang, Yuliy Schwartzburg, and Mark Pauly. Exploring local modifications for constrained meshes. *Computer Graphics Forum*, 32(2pt1):11–20, 2013

- Bailin Deng, Sofien Bouaziz, Mario Deuss, Alexandre Kaspar, Yuliy Schwartzburg, and Mark Pauly. Interactive design exploration for constrained meshes. *Computer-Aided Design*, 2014

- Cheryl Lau, Yuliy Schwartzburg, Appu Shaji, Zahra Sadeghipoor, and Sabine Süsstrunk. Creating personalized jigsaw puzzles. In *Proc. 12th International Symposium on Non-Photorealistic Animation and Rendering*, number EPFL-CONF-199960, 2014

Chapter 2

Related Work

Artists and architects are given a difficult task when modeling an object. The problem statement usually unfolds as such: given a set of limitations–imposed by the software, defined by the realities of the project, subject to physics—sculpt a virtual object to meet aesthetic and physical requirements. The status quo requires checking manually after the design is finished, then following a cycle of drafts and resubmits until the ends are met with certain compromises by the designer. The field of computer graphics is in a perfect position to solve these problems as it is a question of designing software systems using mathematical models and efficient algorithms. Besides professional applications, as CNC machining becomes cheaper and more accessible, similar research can also open up design options to hobbyists and laypersons. The work presented in this thesis is a subset of two larger subfields of computer graphics, architectural geometry and digital fabrication, and falling slightly within the two, computational caustics.

2.1 Architectural Geometry

Mesh Deformation A current trend in architectural design is organic free-form structures that are impossible to construct without computation and are therefore defined parametrically. However, pure generative design must be fine-tuned, reinterpreted, and remodeled with conventional techniques in order to result in a working structure. An open problem is modifying these structures to get a desired outcome based on real-world constraints while keeping with the intentions of the designer. Recently, there has been great interest in the research community in this domain. Eigensatz et al. [EP09] propose a system to deform meshes based on constraints which could be used for performative goals. Shape-Up is a system to interactively explore constrained meshes with geometric

constraints by using handles (also called anchors) [BDS+12]. This is used to satisfy geometric constraints while trading off against smoothness of the mesh. It was extended to support hard constraints as well [DBD+14]. Examples of constraints are planar faces, square faces, angle bounds between edges, prescribed angle lengths, conformal deformation, and many more. Bouaziz et al. extend this to physics constraints, which is useful for visualizing results in deformation or modelling hanging chain models for example [BML+14]. Deng et al. [DBD+13] propose a system to keep such deformations local. Yang et al. [YYPM11] construct a shape space to explore constrained meshes using a 2D interface. Tang et al. [TSG+14] propose to use guided projection rather than optimization to quickly explore hard-constrained meshes.

Construction Other work in architectural design focuses on rationalization given a particular manufacturing technology, which generally implies geometric or performative constraints. Optimizing for planarity, for example, is useful to construct buildings from planar glass. Besides Shape-Up, numerous other works focus on planarization of faces on a mesh [Vax12, POG13]. Other work designs for the topology of a mesh to have planar faces [WLY+08, ZSW10]. Eigensatz et al. [EKS+10, EDS+10] extend the facets to other shapes and present an algorithm to panel free-form surfaces while minimizing production costs by trying to optimize the shape to allow for repeated use of panel molds. Deng et al. [DPW11] present a system to construct surfaces using hexagonal webs which can be planar, geodesic, or circular. Song et al. [SFG+13] design reciprocal frame structures from meshes. We also refer to a recent overview of the connection between geometry and architecture [Pot13].

Several recent papers focus on designing self-supporting surfaces which are masonry structures that can stand with only compression force, requiring no external support [VHWP12, WSW+12, PBSH13, DGAO+13, LPS+13, TSG+14].

2.2 Digital Fabrication

The emerging field of digital fabrication takes advantage of new generation of CNC machines that are coming out on the market and accessible to amateurs and hobbyists. These machines include 3D printers made by companies such as MakerBot® and 3D Systems®, as well as laser cutters made by companies such as Universal Laser Systems® and VersaLASER®.

Shape abstraction Approximating a 3D shape by planar pieces has been studied in the context of geometry processing and rendering. Variational shape approximation [CSAD04] uses Lloyd clustering to distribute a collection of plane proxies over a surface to best approximate a given 3D shape. Billboard clouds [DDSD03] provide highly efficient rendering by optimizing for a small set of textured planes that provide a faithful visual representation of the 3D object. Slices [MSM11] use planar contours for creating shape abstractions guided by perceptual cues derived from a user study. While using the same geometric primitives as we do to describe a shape, these methods are not intended as design tools for physical models, but rather focus on geometric or visual approximation of purely digital representations.

Paper craft Various forms of paper craft modeling have been investigated in computer graphics research. Origami, the art of paper folding without introducing cuts or using glue, has been studied extensively in recent years [DO07, Tac10]. Hart [Har07] introduced modular kirigami, a technique to create intricate symmetric structures from identical paper pieces. Miller and Akleman [MA08] built upon this work and propose a recipe for creating slide-together paper sculptures based on a polyhedral base surface. The constructions focus on specific classes of shapes, such as Platonic or Archimedean solids, and operate on a surface representation, not a 3D volume. Other surface-based methods have been proposed to generate paper models from 3D input geometry, e.g. [MS04, STL06, MGE07]. These algorithms typically flatten a surface and compute a segmentation into developable patches that can be cut from paper sheets and glued together to form a continuous approximation of the given input mesh. Pop-up designs [Gla02, LSH+10, LJGH11] aim at converting a 3D scene into a stable representation supported by two base planes that can be folded flat without stretching or bending the paper.

Laser cutting Laser cutting, printing stencils, and other techniques, can be used with previous methods in computer graphics to fabricate cloth for clothing and toys. Igarashi et al proposed a system to make plush toys out of 3D models [MI07, II08]. Igarashi et al. made it possible to knit 3D models from thread [IIS08]. Another design system makes beadwork from 3D models [IIM12]. More recently, Umetani et al. were able to model and edit clothing in both 2D and 3D while simultaneously seeing what the draped result looks like on a virtual dress form. By using similar methods of fabrication, Skouras et al. propose a system to design inflatable structures like balloons [STK+14] from input 3D models. Garg et al. design a method through which sculptures can be fabricated out of wire meshes by using a scaffold that is made using laser cutting [GSFD+14]. All these methods enable home users to design sculptures from various materials through

the use of computational models, often enabling designs that were previously infeasible.

Chen et al. [CSaLM13] are able to design 3D models out of laser cut pieces, although they focus on the surface rather than the interior. Sliceforms makes 3D models out of laser cut pieces in a grid shape, which can then be flat-folded [LNLRL13]. Mueller et al. [MLB12] use laser cutting for an interactive system to construct mechanical devices. LaserOrigami is another interactive system that allows for laser-cutting 3D objects by allowing the planar slices to bend [MKB13]. NatCut is an interactive editor to fabricate 3D objects from bent laser cut pieces [SSSD+14]. Compared to the work in this thesis, these latter systems lose structural integrity from the holes needed for the bending, and cannot be scaled to medium/large constructions such as furniture.

Closely related to the work in this thesis, Hildebrand and coworkers [HBA12] introduced an algorithm for creating cardboard models based on sliding planar pieces. Their approach focuses on an automatic process that successively adds one element at a time while observing assembly constraints implicitly. In contrast, our system is designed as a dynamic design tool that allows the user to adjust and modify all pieces during the design. Instead of a static data structure that is built incrementally, we propose a continuous optimization that minimally alters the entire configuration to satisfy the constraints. By not limiting the data structure to sequential trees, our work allows for structures with cycles with non-trivial assemblies and assembly orders (such as cases when multiple pieces have to be moved at the same time during assembly, see also the supplementary materials). Additionally, [HBA12] does not consider the angles between planar pieces, keeping them fixed to 90 degrees.

Furniture design Lau et al. [LOMI11] propose an algorithm for generating physical realizations of man-made objects. They introduce a formal grammar for furniture models and combine lexical and structural analysis to automatically create parts and connectors for fabrication. Umetani et al. [UIM12] allow for a guided exploration through physically valid furniture designs. Recently, Shulz et al. use databases of furniture parts to let users design new furniture models out of pre-existing parts [SSLP14]. Agrawala et al. [APH+03] generate step-by-step assembly instructions for models including furniture.

3D printing 3D printing, while seemingly having less constraints than other fabrication methods, involves optimization as well for support structures and printing time. Stava et al. [SVB+12] propose to fix problematic thin and structurally unsound parts of 3D models to be able to print a strong 3D model. Wang et al. [WWY+13] use a

minimal amount of struts to add stability to a structure while fully printing the skin. Build-to-last is a system that optimizes the strength to material ratio of 3D printed objects [LSZ+14]. Dumas et al. [DHL14] propose a system to use bridges for scaffolding in 3D printing. Luo et al. [LBRM12] build larger 3D models out of smaller 3D printed parts. Spec2Fab and Openfab build systems to translate specifications to multi-material 3D printing [CLD+13, VWRKM13].

Fabrication-aware design Digital design tools that incorporate fabrication constraints are becoming popular in mechanical engineering, product design, and architecture [Kil06]. Recently, commercial tools have become available, such as Autodesk 123D, that aim at making fabrication-aware design accessible to non-professional users. Spatial Sketch [WLMI10] is a system that takes sketching input from an infrared light pen, transforms the line drawings into a solid shape, and approximates that shape with a collection of radial or parallel planes. SketchChair [SLMI11] combines intuitive user interfaces, simulation, and automated fabrication to enable personalized chair design. The system generates a set of planar pieces that can be slit into each other to create the framing for the chair, using a predefined grid pattern to specify the combinatorial structure of the chair.

Mechanical Models Digital fabrication is not only useful for static models, but can be used for moving models as well. Calì et al. [CCA+12] print 3D models with joints already assembled. Bächer et al. [BBJP12] fabricate characters that have joints along their medial axis that can be articulated. Zhu et al. [ZXS+12] model mechanical toys that perform simple movements with gearboxes. Ceylan et al. [CLM+13] are able to fabricate mechanical models that illustrate motion capture sequences. Coros et al. [CTN+13] present a system to design moving mechanical characters with gear trains. Thomaszewski et al. [TCG+14] then extend this to linkages.

Assembly Constraints on assembly are the focus of methods for generating geometric puzzles, such as Polyomino puzzles [LFL09] or Burr puzzles [XLF+11]. Séquin et al. [Séq12] deal with multi-hand assemblies of planar pieces. Deuss et al. [DPW+14] deal with the problem of assembling self-supporting structures using chains.

Zimmer et al. [ZLAK14] use simulated annealing to explore the spaces of models that can be constructed using Zometool, a common construction system.

Looking specifically at assembly with LEGO bricks, the LEGO Group themselves pro-

posed the problem of finding assembly instructions for arbitrary models. A first attempt was done by a group of mathematicians in a one week workshop [GHP98]. They created a cost function for a simulated annealing technique but they did not implement or test it. The next attempt used evolutionary algorithms with a cost function inspired from [GHP98]. The results describe the performance of the evolutionary algorithm itself (number of generations before convergence, etc...) but they do not report whether this method can actually make a LEGO model that is constructable [Pet01]. Moreover, the required time that they report for optimization is 5 to 11 hours. Another attempt was done using the beam search technique but there is no data about experimental results [Win05]. Van Zijl and Smal compare existing approaches and propose another approach based on cellular automata [vZS08]. They use a similar merge/split formulation but use several heuristics rather than a graph connectivity formulation. The reported results have a long optimization time of about a few minutes and there is no information on the solidity of the model. All the above methods are based on the cost function proposed by [GHP98]. [SPC09] looks into how to transform a mesh into a LEGO representation but this is only meant for realistic 3D rendering and does not aim at making the model buildable.

2.3 Computational Caustics

Caustics are the result of light being refracted or reflected through an object (see Figure 5.2). Since, at its core, computer graphics focused on the interplay between light and objects, caustics can be found throughout the literature [Gla89]. Traditionally, however, research was focused on the forward problem—given a scene and a light source, render the resulting caustics [PKK00, WS03, GWS04]. We seek to solve the inverse problem—given a target caustic image, find the geometry that will generate such a caustic.

We briefly review several papers that are most closely related to our work in computational caustics. For a broader overview, we refer to the survey of Patow and Pueyo [PP05] that provides an extensive discussion of methods for inverse surface design based on a desired light transport behavior. An interesting related survey focusing on computational tools to design and fabricate material appearance has been presented by Hullin et al. [HIH+13] Dorsch et al. [DLS94] notably are able to achieve two images at two different planes for holograms.

One of the first methods in computer graphics to approach the inverse problem is the work of Finck and colleagues [FDL10]. They use an analysis-by-synthesis strategy that stochastically perturbs a given input surface to better match the desired caustic image.

While interesting results can be achieved with this approach, the optimization, even with an efficient GPU implementation, incurs a high computational cost and can be prone to local minima leading to undesirable artifacts in the generated image. Recently, there have been a number of papers related to our work that apply a more direct optimization approach. These can be divided into discrete patch-based approaches, and continuous parameterization-based methods.

Weyrich et al. [WPMR09] generate a set of sloped, planar microfacets to realize a desired distribution of given ray directions. These microfactes are arranged in a regular grid using simulated annealing to minimize discontinuities. This approach is not designed to reproduce smooth distributions and does not scale well to high resolutions, which limits its applicability for caustic design. Papas et al. [PJJ+11] extend the microfacet approach to curved micropatches, which produce specks of light with an anisotropic Gaussian distribution. While significantly improving the quality of the caustic images, this method retains some of the discretization artifacts and has difficulties in resolving low intensity regions. This is mostly due to using a microfacet array rather than a continuous surface. They model the target image distribution as discrete Gaussian kernels and use capacity-constrained Voronoi tesselations for discretization. We model the target exactly and use Lloyd iterations to initialize a power diagram as discretization.

Yue et al. [YIC+12] propose a different optimization approach using re-configurable prismatic sticks that refract parallel light onto a screen. Their focus is on creative applications where several discrete images can be generated with a single set of refractive sticks. Beyond the limited resolution and other visual deficiencies, a main difficulty with these discrete approaches is that they need to solve a complex spatial arrangement problem. These typically leads to a NP-hard optimizations that require approximation solvers.

Yue et al. [YIC+14] and Kiser et al. [KEN+12] address this problem by formulating a continuous optimization. Their solutions can produce smooth surfaces that lead to a significant improvement in the quality of the caustic images. The core idea is to find a continuous bijective mapping between incoming light and caustic image through a 2D parameterization that locally adapts area to match the desired target brightness. However, enforcing smoothness globally and requiring a one-to-one mapping is unnecessarily restrictive and thus limits the type of caustic images that can be produced. High-contrast regions, intense concentrations of light, or completely black areas are very difficult to achieve this way, as they lead to large deformations of the underlying geometry (see Figure 5.14). In contrast, our algorithm alleviates these difficulties by automatically introducing discontinuities in the normal field at optimally chosen lo-

cations. In addition, our mapping is not constrained to be bijective, enabling intense concentrations of light onto singular points or curves.

Our work is also related to research in computational optics. Glimm and Oliker [GO03] investigated inverse reflector design for far-field distributions, where only the direction of a reflected ray is considered. They showed that certain far-field reflectors can be modeled by a 2nd order nonlinear elliptic PDE of Monge-Ampere type, which can be solved in variational form using Monge-Kantorovich mass transfer. This work (also see [Oli13]), while using very different assumptions and constraints than our solution, inspired our use of an optimal transport formulation.

Rubinstein and Wolansky [RW07] establish a connection to the Klein-Gordon equation to formulate an optimization for intensity control of collimated light beams using a freeform lens. They optimize over both refractive surfaces, but require both the input and output rays to be parallel, thus strongly limiting the contrast of the obtained images.

There have been numerous examples of other work in optics on this topic but they have mostly focused on achieving uniform illumination or a smooth target at low resolution. Ries and Muschaweck [RM02] present a method for tailoring freeform optics and include an integrability condition, although they focus on a point source and their method is inefficient. Magarill [Mag13] presents a fast implementation of Oliker's method for reflectors. Heßling et al. [HGH12] extend the reflectors to extended light sources. Benítez et al. [BMD+04] are able to optimize for several surfaces at once. Wang et al. [WQL07] used a discontinuous lens to prescribe high contrast caustics but their surface cannot be physically realized and is very low resolution. Feng et al. and Bäurle et al. use two surfaces to optimize both for irradiance and wavefront for laser applications as well as to improve the quality of the irradiance distribution [FHJG13, FHGJ13, BBW+12]. To our knowledge, our work is the first to achieve high quality, high contrast, and truly high resolution.

Numerous other works investigate the design and fabrication of freeform optical systems with applications in energy, lasers, illumination, or biomedical imaging, for example. We refer to Fang et al. [FZW+13] for a survey of recent developments. Florian Fournier [Fou10] provides another current example of reflector design in the optics community using Oliker's ellipses method with a faster implementation and also considers extended light sources [FCR09]. In general, the objectives of these methods are substantially different from the ones presented in this thesis, ranging from aberration correction, reflector or lens design for light sources such as LEDs, to solar concentration. In contrast, our work aims at precise control of visually complex caustic projection images with a focus on applications in design, art, and architecture.

Chapter 3

Fabrication-aware Design with Intersecting Planar Pieces

Figure 3.1: 3D designs composed of planar intersecting pieces fabricated using laser cutting and CNC milling.

We propose a computational design approach to generate 3D models composed of interlocking planar pieces. We show how intricate 3D forms can be created by sliding the pieces into each other along straight slits, leading to a simple construction that does not require glue, screws, or other means of support. To facilitate the design process, we present an abstraction model that formalizes the main geometric constraints imposed by fabrication and assembly, and incorporates conditions on the rigidity of the

resulting structure. We show that the tight coupling of constraints makes manual design highly nontrivial and introduce an optimization method to automate constraint satisfaction based on an analysis of the constraint relation graph. This algorithm ensures that the planar parts can be fabricated and assembled. We demonstrate the versatility of our approach by creating 3D toy models, an architectural design study, and several examples of functional furniture.

3.1 Foreword

3D constructions composed of interlocking planar pieces (Figure 3.1) are popular for creating 3D toys made from wood or cardboard, but are also of great interest in architecture and interior design (Figure 3.2). The popularity of these models is mainly due to the ease of fabrication and assembly. Planar pieces can be cut easily and cheaply from many different materials, including cardboard, wood, metal, plastic, glass, or stone, using simple machinery such as saws or laser cutters. Compared to 3D printing, 2D cutting not only allows for a wide variety of materials, but also a much larger range of scale, enabling constructions of the size of buildings at comparatively low cost.

In this chapter we define geometric constraints for 3D objects composed of planar interlocking pieces and propose an optimization approach that solves for a physically realizable solution for a given user input of planar piece orientations, positions, and outlines. A feasible solution satisfies geometric conditions that directly relate to fabrication, stability, and assembly.

If two pieces intersect this introduces an angle constraint on the plane normals. We derive sufficient conditions on the relation of these constraints that guarantee rigidity of the final assembled structure. To guarantee a valid assembly sequence, each pair of intersecting planar pieces is constrained to be displaced only in the direction of the slit created along their intersection line. These slit constraints can easily create a locked state in which a subset of the pieces cannot be moved at all. Our optimization automatically avoids such configurations and ensures that the 3D design can be assembled. Beyond the orientation and placement of planar pieces, we also optimize their geometric shape to avoid collisions during assembly.

The space defined by angle and slit constraints can quickly become difficult to navigate in, since cycles in the graph can introduce complex non-local dependencies. Modifications of the orientation of a single piece to satisfy a certain design intent can propagate through the entire object, making manual control of all constraints cumbersome and

Taxidermy | Cardboard Furniture | Wooden Dinosaur

Acrylic Sculpture | Metropol Parasol, Sevilla, Spain

Figure 3.2: Intersecting planar pieces can lead to compelling 3D forms with applications in interior design, toy making, sculptural art, and architecture.

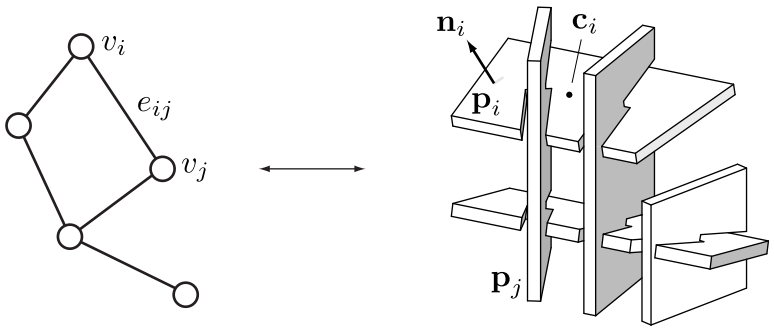
virtually impossible for more advanced designs. This complexity is reflected in the fact that existing manual designs are often limited to non-cyclic graphs or grid-like structures (see Figure 3.2). We show that these restrictions are not necessary and introduce a more flexible constraint space that enables a variety of designs that cannot be achieved by current methods.

Contributions. The input to our system is a set of initial plane orientations and positions given by the user or created by some generative procedure. These can be specified at the start or during the design process in any order. Planar piece outlines are either defined explicitly or found by intersecting an infinite plane with a 3D hull. During interactive editing, the designer can freely modify the position or orientation of planar pieces, add or remove pieces, or change the constraint graph. The solution is automatically updated to ensure that all constraints are satisfied.

The output is a set of 2D stencil curves and a corresponding assembly plan. The curves can be transformed to machining instructions and sent to standard planar cutting machines. The fabricated planar pieces can be flat-packed efficiently and assembled without glue or screws. We believe that the simplicity, flexibility, and low cost of this type of construction makes our system interesting both for casual users who want to create their own 3D toys or freeform furniture, as well as architects and designers who want to realize ambitious large-scale productions.

3.2 Problem Formulation

We first introduce some notation to provide a formal description of our optimization and design objectives. In Section 3.3, we will discuss a method to realize these objectives. We represent an arrangement of intersecting planar pieces by a graph $G = (\mathcal{V}, \mathcal{E})$ consisting of a set $\mathcal{V} = \{v_1, \ldots, v_n\}$ of n nodes or vertices, and a set $\mathcal{E} = \{e_1, \ldots, e_m\}$ of m edges with $e_k \in \mathcal{V} \times \mathcal{V}$. Each node v_i represents a planar piece $\mathbf{p}_i$ with a normal $\mathbf{n}_i$, and the centroid of the piece $\mathbf{c}_i$ which lies on the plane (see illustration below). We write e_{ij}



constraint graph planar intersecting pieces

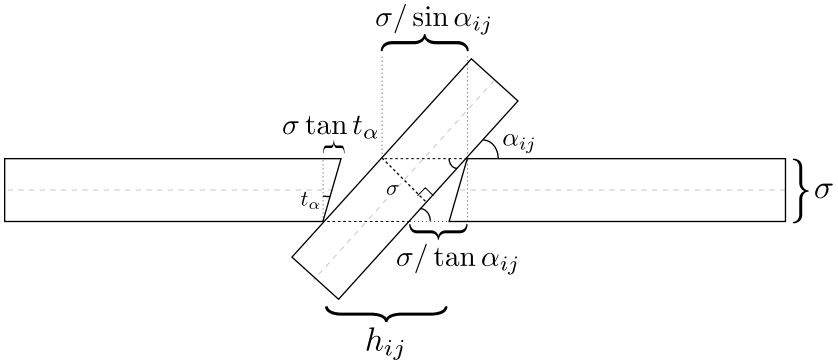
to denote an edge that connects the vertices v_i and v_j. An edge $e_{ij} \in \mathcal{E}$ defines a slit connection between two nodes v_i and v_j. We denote with $\alpha_{ij} = \arccos(|\mathbf{n}_i \cdot \mathbf{n}_j|)$ the corresponding intersection angle of the two plane normals $\mathbf{n}_i$ and $\mathbf{n}_j$, where $\alpha_{ij} \in [0, \pi/2]$ always measures the smaller of the two intersection angles.

3.2.1 Rigidity

To ensure stability of the structure, we must introduce constraints on the angles at which the pieces intersect. These angles restrict possible rotations (see Figure 3.3(e)) and contribute to the stress at the intersection slit. The angle at which a slit can be cut

into a planar piece is constrained by machining limitations. Many tools (e.g. most laser cutters) can only cut a planar piece in a direction orthogonal to the plane normal, while other machinery (e.g. CNC milling devices or 3D laser cutters) have bounds on how far they can deviate from orthogonality (typically by about 40 degrees for a five-axis milling machine). We denote the maximum possible cutting angle relative to the plane normal as $t_\alpha \in [0, \pi/2)$. This cutting angle has direct consequences on the width h_{ij} of the slits: Let σ be the thickness of the material. Then for an edge e_{ij} with corresponding intersection angle α_{ij} the minimal required slit width is

$$h_{ij} = \sigma / \sin \alpha_{ij} + \max(\sigma / \tan \alpha_{ij} - \sigma \tan t_\alpha, 0). \tag{3.1}$$



Angle constraints. We call a slit *tight*, if the pieces are touching along the entire intersection, or $\sigma \tan t_\alpha \geq \sigma / \tan \alpha_{ij}$. It follows that the slit of edge e_{ij} can only be tight if $\alpha_{ij} \geq \pi/2 - t_\alpha$. We call this constraint on the intersection of two planar pieces an *angle constraint*. In our optimization we aim for tight slits as these provide the most stable configurations. In particular for dense graphs, however, it is not always possible to obtain large enough intersection angles to achieve tight slits everywhere. Fortunately, we can achieve stable configurations even when widening the slit width.

We say a node $v_i \in \mathcal{V}$ is *rigid*, if the corresponding piece has no free rotational motion space, i.e. if the slit connections with other intersecting nodes prevent any rotational motion of the piece with respect to its adjacent pieces. We call a graph G *locally rigid*, if all nodes $v_i \in \mathcal{V}$ are rigid. As illustrated in Figure 3.3, tight slits trivially ensure rigidity of a piece. Alternatively, several slits can work together to eliminate any free rotational motion, thus ensuring rigidity even in the absence of tight slits. For a single piece, we can see that any two (non-collinear) slits already guarantee rigidity if the two connected pieces are themselves fixed, as the piece cannot rotate without pushing on the connected pieces (see Figure 3.4(a)).

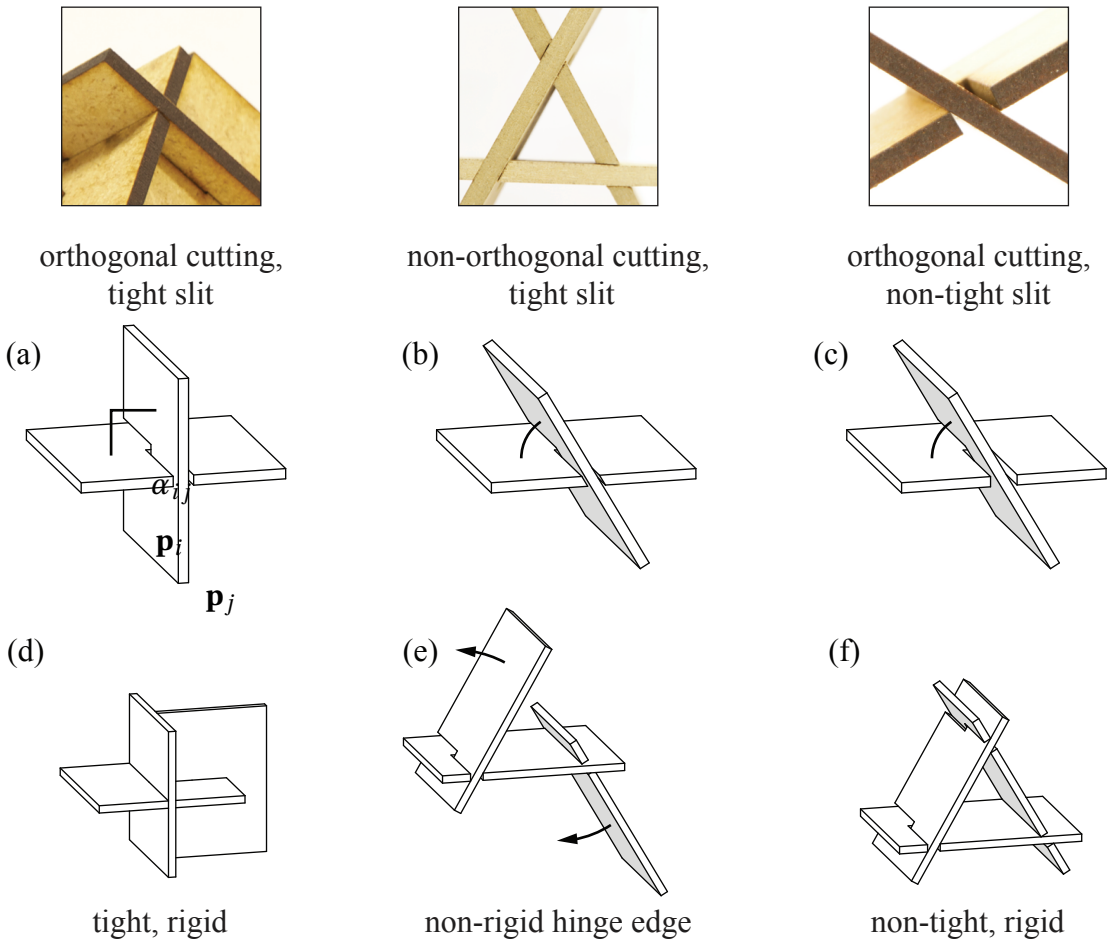


Figure 3.3: Machining restrictions impose constraints on the maximal cutting angle of slits. Most devices (e.g. laser cutters) can only produce cuts orthogonal to the surface plane (a, c), while others (e.g. 5-axis milling) can accommodate slits up to some angular threshold (b). Tight slits (a, b, d) are generally preferable in terms of stability, but can significantly restrict the possible intersection angle of pieces. Wide slits (c) allow more freedom, but can introduce undesirable hinge edges that might lead to instabilities (e). Even when resorting to wide slits, our optimization is guaranteed to find a locally rigid configuration as illustrated in (f).

Non-tight edges that are not part of a cycle of a graph will allow rotational motion around the slits. We call these *hinge edges* and write $\mathscr{E}^h \subseteq \mathscr{E}$ for the set of all hinge edges. Therefore, a necessary (but not sufficient) condition for local rigidity is that the slits of all hinge edges must be tight, i.e. $\alpha_{ij} \geq \pi/2 - t_\alpha \; \forall e_{ij} \in \mathscr{E}^h$.

Global rigidity, or preventing rotation in any cycle of G and not just per piece, is a difficult problem related to the rigidity of graphs [Jac07]. A non-globally rigid graph

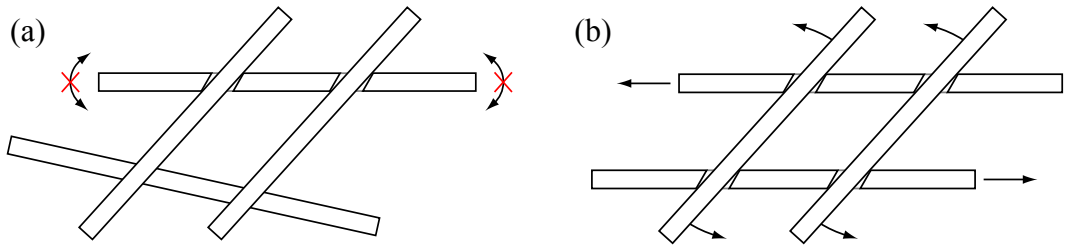


Figure 3.4: A piece with non-tight slits (a) can still be rigid if its neighboring pieces are fixed. A configuration with four parallel non-tight slits in a cycle (b) is locally but not globally rigid. Therefore, our optimization always aims for tight slits if possible.

is shown in Figure 3.4(b). Rotation involving multiple pieces, as illustrated here, is unlikely to occur without specific initial conditions. Even so, to prevent this, we aim for tight slits wherever possible. Configurations with wide slits can also be less desirable for physical realization, since forces acting on these slits will be concentrated on the contact edges, which might lead to strong internal stresses and corresponding material wear [Dow93]. We call a graph *strongly rigid*, if all slits are tight, i.e. $\alpha_{ij} \geq \pi/2 - t_\alpha \; \forall e_{ij} \in \mathscr{E}$. Our optimization always tries to find a strongly rigid solution first. Only if such a solution cannot be obtained does the algorithm resort to wider slits, while still ensuring that the configuration remains locally rigid. If there is no feasible tight solution or if a tight solution is not desired, coupled rotations and material stress could be checked afterwards with a physics simulation in a similar manner to [UIM12].

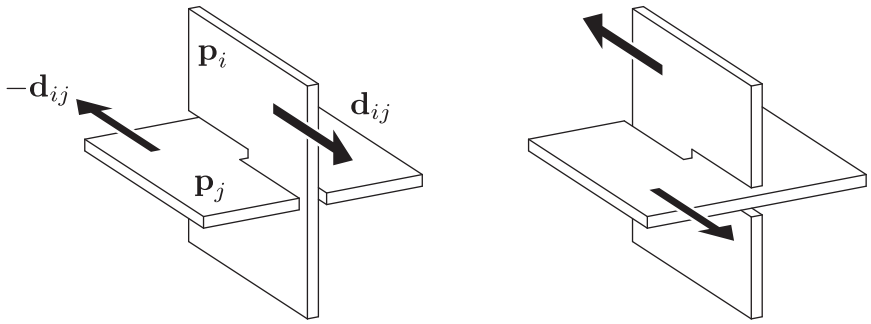
3.2.2 Assembly

If we introduce cycles in the graph, similar to the cycle in Figure 3.5, then it can become impossible to separate the pieces and likewise impossible to assemble the object. Therefore, we consider the problem of disassembly: If a 3D object represented by a graph G can be separated into individual pieces, then the inversion of this disassembly sequence yields an assembly plan. In this section, we focus on how to define and detect if a structure is assemblable. We describe a solution to ensure this constraint for an invalid configuration in Section 3.3.1.

We need to satisfy two types of constraints to guarantee existence of a disassembly sequence: (i) Slit constraints and (ii) global collision constraints. Slit constraints restrict the relative motion for separating two interlocking pieces and are independent of the actual shape of the pieces. Effectively, these constraints are due to the collisions of the

two pieces in every direction but that of the slit and the coupling of these collisions (see Figure 3.5). They therefore depend only on the intersection direction $\mathbf{d}_{ij}$ and the structure of the constraint graph. Global collisions, or ensuring collision-free paths during assembly, are discussed in Section 3.4.

Slit constraints. The physical connection defined by an edge $e_{ij} \in \mathcal{E}$ is realized by introducing straight slits into the planar pieces $\mathbf{p}_i$ and $\mathbf{p}_j$ (Figure 3.3). We assume that $\mathbf{p}_i$ and $\mathbf{p}_j$ can only be separated by translating them in opposite directions along the vector $\mathbf{n}_i \times \mathbf{n}_j$. We assume this is the only way to separate a connection, i.e. no rotation or deformation of a piece is allowed or necessary. Since both configurations



are generally possible for a slit (see inset illustration), we define the edge direction vector as $\mathbf{d}_{ij} = \pm\mathbf{n}_i \times \mathbf{n}_j$, using the sign convention that piece $\mathbf{p}_i$ can only be moved in the direction $\mathbf{d}_{ij}$ relative to $\mathbf{p}_j$, and, analogously, $\mathbf{p}_j$ can only be moved in direction $-\mathbf{d}_{ij}$ relative to $\mathbf{p}_i$. We call these restrictions on the relative motion of pieces $\mathbf{p}_i$ and $\mathbf{p}_j$ the *slit constraint* of edge e_{ij}.

Assembly conditions. We call two edges e_{ij} and e_{kl} *parallel*, if $\mathbf{d}_{ij} = \mathbf{d}_{kl}$. We say a graph $G = (\mathcal{V}, \mathcal{E})$ can be *split* into two subgraphs G_1 and G_2, if there exists

- a cut into two non-empty vertex sets $\mathcal{V}_1$ and $\mathcal{V}_2$ with $\mathcal{V}_1 \cup \mathcal{V}_2 = \mathcal{V}$ and $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$, and
- a cut-set of parallel edges $\mathcal{E}' \subseteq \mathcal{E}$ such that $v_i \in \mathcal{V}_1$ and $v_j \in \mathcal{V}_2$ for all $e_{ij} \in \mathcal{E}'$.

The two subgraphs are then given as $G_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $G_2 = (\mathcal{V}_2, \mathcal{E}_2)$, where the edge sets $\mathcal{E}_1$ and $\mathcal{E}_2$ are the restrictions of $\mathcal{E}$ to $\mathcal{V}_1$ and $\mathcal{V}_2$, respectively. We call the edge set $\mathcal{E}'$ a *parallel cut* (as opposed to other cuts in the graph that do not consist of only parallel edges). The existence of a parallel cut means that we can displace all pieces in $\mathcal{V}_1$ along the common edge direction of $\mathcal{E}'$ to separate them from the pieces in $\mathcal{V}_2$ without violating any of the slit constraints.

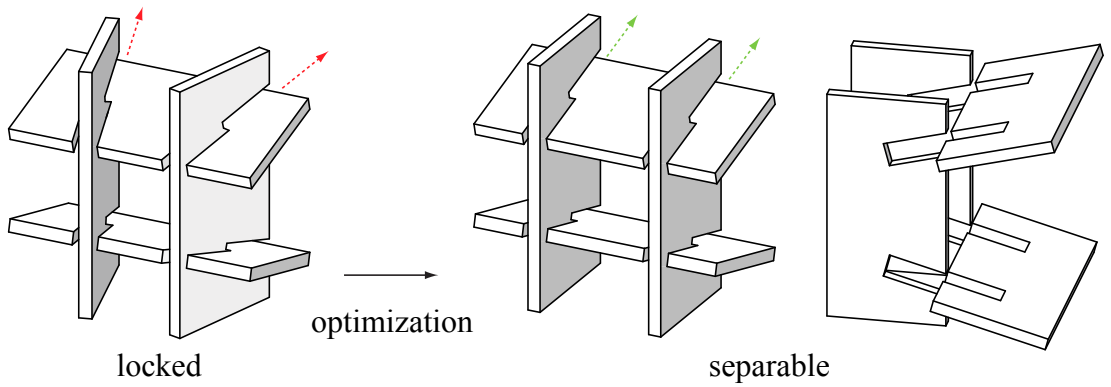


Figure 3.5: Planar intersecting pieces can easily create locked configurations (left) in which no piece can be moved and hence the structure cannot be assembled. Our algorithm optimizes for plane orientations that ensure feasibility of assembly (right).

Definition I: We call a graph $G = (\mathcal{V}, \mathcal{E})$ *separable*, if

- $|\mathcal{V}| = 1$, or
- G can be split into two separable subgraphs G_1 and G_2.

It follows immediately from Definition I that any graph that does not contain cycles, i.e. is a tree, is separable, since any edge defines a parallel cut-set. Cycles in the graph, however, require the existence of a set of parallel edges that cuts the cycle into two or more parts.

Algorithm 1 ISSEPARABLE($G = (\mathcal{V}, \mathcal{E})$)

```
if |V| = 1 then return true
else
    for do e ∈ E
        // find all edges parallel to e
        E' ← {e' ∈ E | d(e') = d(e)}
        // if parallel cut exists, cut graph and recurse
        if then E' contains a cut C ⊆ E' of G
            split G along C into G1 and G2 return ISSEPARABLE (G1) & ISSEPARABLE (G2)
        end if
    end for return false
end if
```

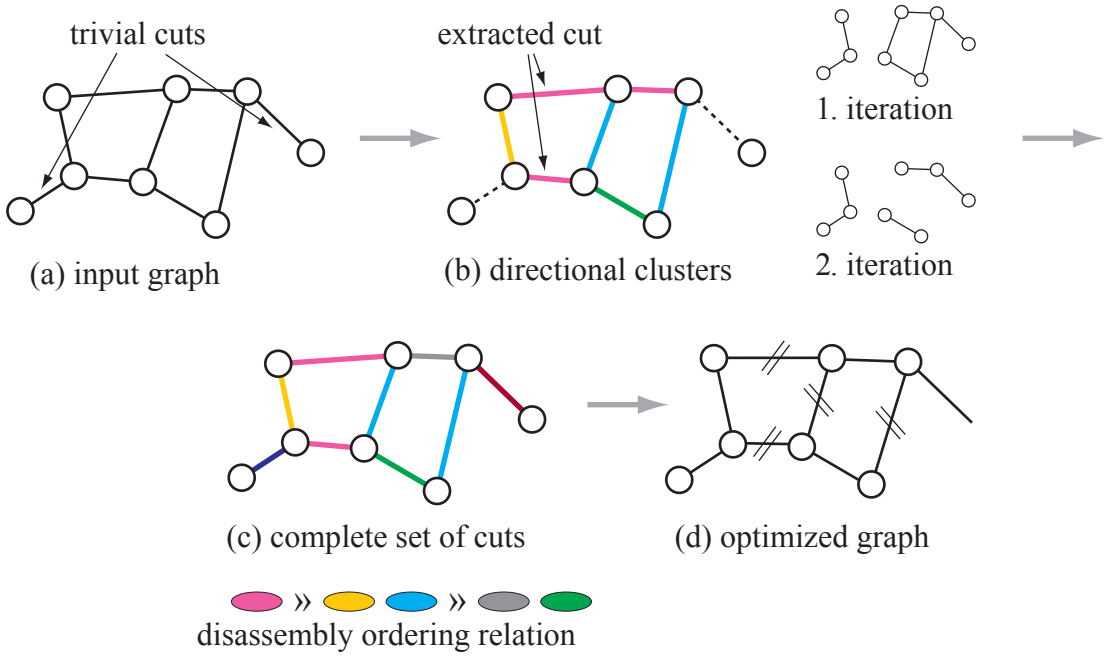


Figure 3.6: Optimizing for assembly. Ignoring trivial cuts, edges are clustered according to the slit direction (illustrated here by the 2D graph edge direction) and a cut is extracted from the clusters (b). The corresponding edges are removed and clustering is repeated until a complete set of cuts (indicated by color) is obtained (c). The optimization then ensures that all edges of the same cut become parallel (d) so that the resulting object can be disassembled. The sequence of disassembly is constrained by the partial ordering relation resulting from the clustering method. The dark purple and red connections can be separated at any time and are thus not part of the ordering relation.

From Definition I we can derive a recursive algorithm that checks if a given graph G is separable and hence the corresponding 3D structure disassemblable (see Algorithm 1). This algorithm makes use of an important observation: If a graph is separable and contains multiple parallel cuts, then applying any of these cuts in any order will create separable subgraphs since the edges of the other cuts will define parallel cuts in one or more of the new subgraphs as well. Consequently, we can simply iterate through all edges in the edge set $\mathscr{E}$, apply the first parallel cut that we find (if one exists) to split the graph into two, and execute the same procedure recursively on the generated subgraphs. Any cycle in the graph that does not contain a parallel cut is in a locked state, i.e. cannot be disassembled (see Figure 3.5).

3.3 Optimization

Section 3.2.1 and Algorithm I provide a means to test whether a given design graph G is rigid and can be assembled. Our goal is now to create such valid configurations. The complex coupling of constraints makes it imperative to augment the design process with an optimization method. By handling constraint satisfaction automatically, the user is relieved of this difficult task and can focus on high-level aspects of the design.

An inherent problem with optimization is that as the orientations and positions of the pieces change, edges would need to be added or removed from the intersection graph. This quickly leads to intractability. To avoid this, we make use of the observation that the angle and slit constraints only depend on the plane orientations, but not the spatial positioning of planar pieces nor their geometric shape. We can thus formulate the optimization as a two step procedure: Step one solves for a feasible solution using only the plane normal vectors $\mathbf{n}_1,\ldots,\mathbf{n}_n$ as unknowns (Section 3.3.1). Step two then optimizes separately for the centroids $\mathbf{c}_1,\ldots,\mathbf{c}_n$ and the boundaries of each piece to maintain the structure of the graph (Section 3.3.2).

3.3.1 Orientation Optimization

The input for the optimization is an initial arrangement of planes with the corresponding constraint graph $G = (\mathcal{V},\mathcal{E})$ that specifies which planes should be intersecting.

Satisfying angle constraints. To obtain tight slits we define an inequality constraint to force the intersection angles α_{ij} above the threshold mandated by fabrication (Section 3.2.1). For each edge e_{ij}, let $c_{\text{angle}}(e_{ij})$ denote how far the corresponding planar pieces are from satisfying the angle threshold measured as

$$c_{\text{angle}}(e_{ij}) = \left(\mathbf{n}_i \cdot \mathbf{n}_j\right)^2 - \sin^2 t_\alpha. \tag{3.2}$$

We use the square of the dot product to implicitly deal with the two possible orientations. It follows that the graph G is strongly rigid, if $c_{\text{angle}} < 0 \;\forall e_{ij}$.

Satisfying slit constraints. In order to guarantee the existence of an assembly sequence, we need to create a series of parallel cuts in the graph that separate all planar pieces from each other. Our algorithm first finds a complete set of parallel or almost parallel cuts, the latter of which are then optimized to become parallel. The goal here

is to induce minimal change to the current plane orientation to achieve edge parallelism and thus guarantee assembly.

Based on this objective, our algorithm proceeds as follows (see Figure 3.6): We first discard all hinge edges, i.e. edges that are not part of a cycle or collinear edges, since these edges provide trivial parallel cuts and thus do not impose any critical slit constraints. For the remaining edges that belong to a cycle, we perform k-means clustering based on edge direction, starting with k equal to the number of edges. We then progressively decrease k until one of the clusters contains a cut of the graph. To avoid unnecessary modifications to the plane normals when making the cluster parallel, we select the smallest subset of the cluster edges that define the cut. These edges are tagged and assigned to the same cut set for the optimization. We then continue the clustering recursively on the untagged edges of the generated subgraphs until all edges are covered.

This yields a decomposition of the edge set into disjoint sets $\mathcal{S}_1, \ldots, \mathcal{S}_M$, $\mathcal{S}_k \subseteq \mathcal{E}$ that, when applied in sequence, form a complete series of cuts of G. Since these cuts are not necessarily parallel cuts, we align the directions of all edges within each cut set to a common direction to achieve parallelism. We introduce an auxiliary unit vector $\mathbf{s}_k$ for each set $\mathcal{S}_k$ to represent this common constraint direction and formulate equality constraints for each edge in $\mathcal{S}_k$ by quantifying the difference in alignment of the edge direction vectors to the vector $\mathbf{s}_k$. This difference is formulated for each $e_{ij} \in \mathcal{S}_k$ as

$$c_{\text{slit}}(e_{ij}, \mathbf{s}_k) = \|(\mathbf{n}_i \times \mathbf{n}_j) \times \mathbf{s}_k\|^2. \tag{3.3}$$

Given the sequence of cut sets $\mathcal{S}_1, \ldots, \mathcal{S}_M$, it follows that G can be assembled, if $c_{\text{slit}}(e_{ij}, \mathbf{s}_k) = 0 \; \forall \mathcal{S}_k$ and $\forall e_{ij} \in \mathcal{S}_k$.

Satisfying user constraints. Our goal is to find a configuration of plane normals that satisfies all angle and slit constraints, while being as close as possible to the design input of the user. We therefore introduce a closeness objective function $f_{\text{input}}(v_i) = \|\mathbf{n}_i - \mathbf{n}'_i\|_2$ that measures the deviation of the current plane normal to the initial input normal vector $\mathbf{n}'_i$. The corresponding energy term is given as

$$E_{\text{input}} = \sum_{i=1}^{n} \omega_i f_{\text{input}}(v_i)^2, \tag{3.4}$$

where the weights ω_i allow the user to specify important aspects of the design intent. Pieces that should not change substantially can be assigned a high weight ("fixing the piece"), while a low weight can be set for pieces that are free to deviate more strongly from the input configuration.

Constraint optimization. The goal of finding a configuration of planar pieces that is rigid and assemblable can now be formulated as a quadratic objective function with quadratic equality and inequality constraints:

$$\arg \min_{\mathbf{n}_1,\ldots,\mathbf{n}_N} E_{\text{input}}$$

$$\begin{aligned} \text{subject to } & c_{\text{angle}} \leq 0 \;\; \forall \mathbf{e}_{ij}, \\ & c_{\text{slit}} = 0 \;\; \forall \mathscr{S}_k, \;\; \forall \mathbf{e}_{ij} \in \mathscr{S}_k, \\ & \|\mathbf{n}_i\|^2 = 1 \;\; \forall \mathbf{n}_i, \\ & \|\mathbf{s}_i\|^2 = 1 \;\; \forall \mathbf{s}_i. \end{aligned} \tag{3.5}$$

The unknowns are the auxiliary vectors $\mathbf{s}_1, \ldots, \mathbf{s}_M$, and the plane normals $\mathbf{n}_1, \ldots, \mathbf{n}_N$ that determine both the intersection angle of connected planes as well as their slit direction. We also add additional equality constraints to keep the vectors $\mathbf{n}_i$, $\mathbf{s}_i$ normalized. We use Sequential Quadratic Programming (SQP) to solve Equation 3.5.

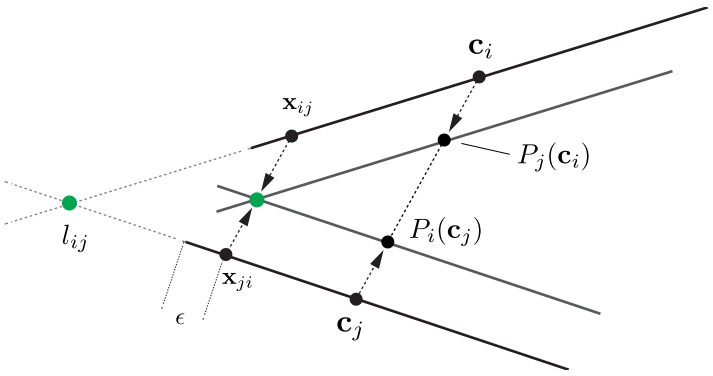
Rigidity. If the optimization defined above yields a solution, the resulting 3D structure is guaranteed to be strongly rigid. If no solution is found, we relax the requirement of strong rigidity to introduce more degrees of freedom by allowing edges that are part of cycles to violate the angle constraint. This will introduce wider slits but still guarantees a locally rigid configuration. We simply select the non-hinge edge with the highest c_{angle}, remove the corresponding angle constraint, and re-run the optimization. We iterate this procedure until a feasible solution is found. Note that the optimization is guaranteed to find such a feasible solution. If all angle constraints are invalidated, strong rigidity is only enforced on the hinge edges. Since these can only be part of a trivial parallel cut, they are not in conflict with the slit constraints. Consequently, a solution always exists (but might be far from the input), for example when all edges become parallel. In this case, the configuration might need to be checked for global rigidity (as in Figure 3.4).

3.3.2 Position Optimization

The optimization described above solves for plane orientations that satisfy assembly and rigidity constraints. Changes in plane orientation for a piece v_i are effected by rotating around the centroids $\mathbf{c}_i$. As a result, some pieces that are connected by an edge in the graph G might no longer be intersecting, while others might now intersect even though there is no corresponding edge in G.

For each newly introduced intersection, if a corresponding edge can be introduced without violating the constraints, we add it to G. Then, since the intersection pattern defined by the edge set $\mathscr{E}$ is closely related to visual or functional semantics of the design, we apply a second optimization step to ensure that the graph structure is preserved. Recall that the $\mathbf{c}_i$ and the boundaries of the pieces have no direct influence on the angle and slit constraints as these only depend on the plane normals. Thus we can, independently of the above orientation optimization, modify the relative positioning of the planar pieces and their contours to maintain the graph intersection pattern. We can also handle collisions during the assembly process in this step.

During the design process, the user can define the contours explicitly or may optionally choose a 3D guiding volume (see for example Figure 3.8). Each piece would then be maximally intersected with the guiding volume to obtain its contours. This choice results in slight differences in the method that will be noted below.



Retaining intersections. Let us assume that the planar pieces v_i and v_j of a given edge e_{ij} no longer intersect after their normals have been modified by the orientation optimization. Let line l_{ij} be the intersection of the infinite planes of $\mathbf{p}_i$ and $\mathbf{p}_j$ (see inset). For $\mathbf{p}_i$, we find the furthest point lying on the piece in the direction of $\mathbf{c}_i$ to its projection on l_{ij} and subtract a small distance ϵ to account for the slit width and allow for stability. We call this point x_{ij}. A similar operation on $\mathbf{p}_j$ gives us $\mathbf{x}_{ji}$. The target points $P_j(\mathbf{c}_i)$ and $P_i(\mathbf{c}_j)$ are such that $\mathbf{x}_{ij}$ and $\mathbf{x}_{ji}$ coincide:

$$P_j(\mathbf{c}_i) = \mathbf{c}_i - \frac{\mathbf{x}_{ji} - \mathbf{x}_{ij}}{2} \tag{3.6}$$

If the pieces intersect then $P_j(\mathbf{c}_i) = \mathbf{c}_i$. If the user specifies a guiding volume, then we additionally project $\mathbf{x}_{ij}$ into the volume to ensure an intersection, again adding an ϵ

tolerance. We formulate the optimization as a minimization of the following energy:

$$E_{\text{pos}} = \sum_{i=1}^{n} \sum_{e_{ij} \in \mathscr{E}} \|\mathbf{c}_i - P_j(\mathbf{c}_i)\|, \tag{3.7}$$

where e_{ij} are the out-edges of v_i. We optimize for all positions simultaneously using a non-linear alternating projection scheme (see [BXM03]), iterating until convergence. The optimization converges to the closest local minimum to the input since each step weakly decreases the distance between each pair of pieces. While it still might be possible in degenerate cases that all $\mathbf{c}_i$ converge to a single point, we never observed this in our experiments and usually 10 iterations is sufficient.

Contouring. The position optimization displaces planar pieces to ensure that all intersections specified in the constraint graph are realized. This operation might lead to new intersections requiring new edges to be added. To avoid these and the resulting additional constraints, we apply a clipping algorithm that iteratively clips pieces, changing only contours, to avoid collisions. This leaves no new intersections, but can result in split pieces and possibly disconnected structures. Since the choice of which planes clip which and in what order can change the piece geometry and aesthetics, we perform only the necessary clipping in order for the collisions to be avoided, and default sensibly in ambiguous cases.

We aim to clip without changing the intersection graph: Say there are two intersecting pieces $\mathbf{p}_i$ and $\mathbf{p}_j$ where $e_{ij} \notin \mathscr{E}$. If the pieces do not intersect along the entire length of one piece, then we can simply clip without changing the graph since the pieces stay connected in relation to the slits (see Figure 3.7(a-b)). Otherwise, we split one piece by another such that the graph does not get disconnected. In ambiguous cases (neither split disconnects the graph or both do) we default to splitting the piece that has no slits on one side of the intersection or the one with most intersections. By clipping one piece with another, we effectively split a vertex in our intersection graph into two vertices, which introduces additional degrees of freedom in our constraint graph. We therefore rerun the orientation optimization to ensure constraint satisfaction (see Figure 3.7(c)).

Global collisions. As opposed to only checking for introduced intersections, we can also look at the global configuration and apply a final step that aims at resolving collisions by cutting pieces if they block the motion path of another piece. This procedure works in the order of disassembly, i.e. we start with the piece that would be taken out first from the assembled object. To simplify the collision analysis, as in [HBA12], we only consider the straight-line motion space defined by the slit direction. Thus every

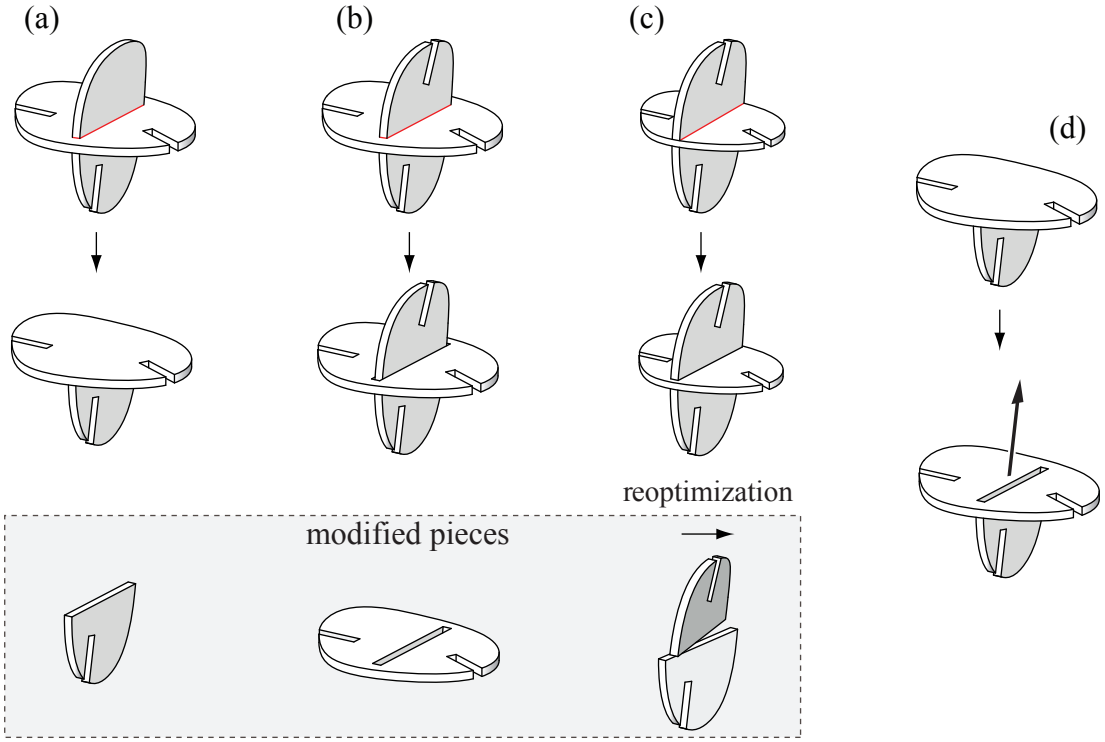


Figure 3.7: After optimization of orientations, intersections may arise that prevent assembly (red lines). The system clips each piece such that no new intersections are introduced. (a) and (b) illustrate clipping without changing the intersection graph. If it is not possible to keep the same graph, we split a piece into two and re-optimize for orientation, as in (c). The bottom row shows only the modified pieces for illustration. Clipping can also be performed such that the assembly path is free of collisions (d). Afterwards, the piece can be taken out in the direction of assembly. Note this will also ensure that clipping of (b) will not introduce assembly constraints.

piece defines a collision volume that can be generated by sweeping the piece geometry in the slit direction. We automatically detect collisions of this volume with all other pieces that have not yet been taken out and clip these pieces to allow full motion (see Figure 3.7(d)). If this splits a piece into two, we continue as described in the previous paragraph. Recall that every slit can be realized in two opposing orientations (see inset illustration in Section 3.2). Therefore, if the graph becomes disconnected, we also have the degree of freedom of choosing the opposite orientation for each parallel cut.

3.4 Design Process

With the optimization method in place, we now introduce an interactive design process that allows the user to create 3D objects composed of intersecting planar pieces. The interactive nature of the design process is shown in Figure 3.14.

Constraint editing. In order to keep the problem tractable, we keep the intersection graph G fixed during a single optimization. However, at any stage during the design, the constraint graph can be modified by the user to explore different design options, adding possible intersections and re-optimizing to see the possible resulting structure. To increase the degrees of freedom the user can discard a piece, i.e. remove a node from the graph, or eliminate an intersection, i.e. delete an edge. One advantage of the clustering method illustrated in Figure 3.6 is that we can simply permute the ranking of the cuts to propose different design solutions to the user. This allows exploring design alternatives by simply browsing through a set of solutions that all satisfy the constraint sets. For the ambiguous cases in Section 3.3.2, the user can choose between possible clippings.

Another handle that is at our disposal for avoiding collisions is the orientation of slits within each parallel cut set. Recall that every slit can be realized in two opposing orientations (see inset illustration in Section 3.2). After the optimization, we make sure that the slits are all facing the same direction for each parallel cut. However, one can freely choose the opposite orientation for each slit, as long as the parallel cuts remain in the same orientation. This will only change the global collision clipping and the user can choose the opposite orientations for aesthetic or static reasons. For example, if the wrong orientation is selected, the pieces might slide out due to gravity or other forces acting on the design.

3.5 Results

We show several designs created with our system to evaluate the versatility and effectiveness of our approach. Figure 3.8 shows the design of a wooden 3D toy based on a guiding surface mesh. This dinosaur toy was designed in about 20 minutes with a concrete idea of the intended structure. Figure 3.9 shows an application in architecture, highlighting the potential of our method for design exploration and the creation of scale models. A feasible construction was ensured with our method with the process finished

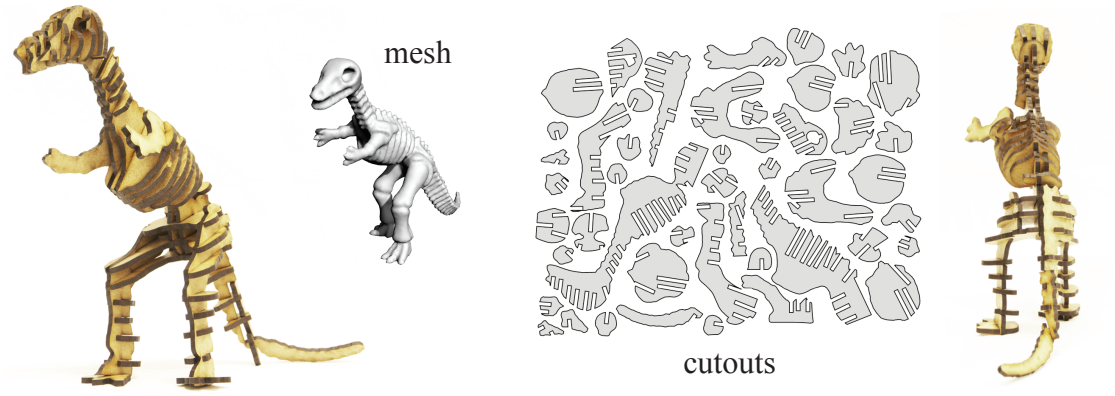


Figure 3.8: A 3D toy composed of orthogonally intersecting planar pieces. The mesh of the dinosaur serves as a guiding volume for the design.

Figure 3.9: An architectural design study of an outdoor pavilion made from orthogonally intersecting pieces.

in less than 15 minutes. Figure 3.10 illustrates how functional custom-designed furniture can be generated with our system. These examples were designed by exploration with only a minimal idea of the intended outcome. Even with a relatively simple geometry, the assembly of this model requires joining multiple pieces at the same time along non-trivial parallel cuts. Existing methods that incrementally add one piece at a time [HBA12] cannot deal with such configurations. The assembly process as prescribed by our framework can be seen in Figure 3.15.

All examples in Figures 3.8 to 3.10 are strongly rigid, i.e. only contain tight slits. Figure 3.11 is an example with non-tight slits that still maintains rigidity. The assembly of such models is in general more complex, since non-tight slits can lead to non-rigid constellations during construction that only become rigid once the corresponding cycles are closed. In return, we obtain a substantially richer design space, in particular when work-

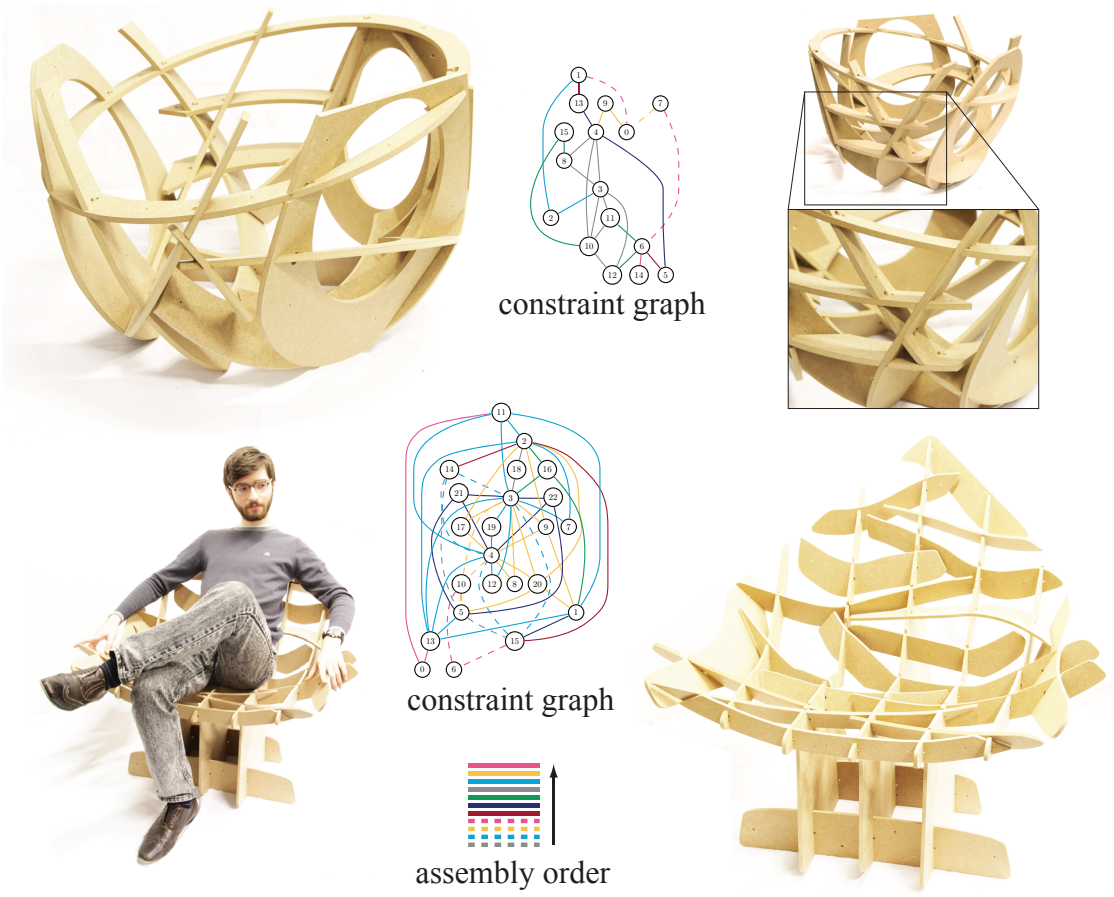


Figure 3.10: The base of a coffee table and a freeform chair milled from medium density fiberboard (MDF) (see also Figure 3.1). Complex joints like the one illustrated in the zoomed image can easily be created with our method. The design graphs illustrate the complex coupling of constraints. The parallel cut sets computed by the optimization are indicated by color in the order of assembly. The smallest intersection angle is 50°, as mandated by the CNC milling machine.

ing with orthogonal cutting devices such as laser cutters. This is an important advantage over previous methods such as [HBA12] that rely on tight slits for rigidity and therefore quickly lead to grid-like structures when dealing with dense constraint graphs. With the chair and table examples, milling took one hour and the assembly of each model required about half an hour. The geometric information created during the design, i.e. the contour curves of the planar pieces, is directly translated into machining instructions via scripting. To facilitate assembly, we can optionally create unique matching IDs for each slit that are laser-etched into the piece. An assembly plan can be created directly from the partial ordering generated by the clustering algorithm of Section 3.3 as illus-

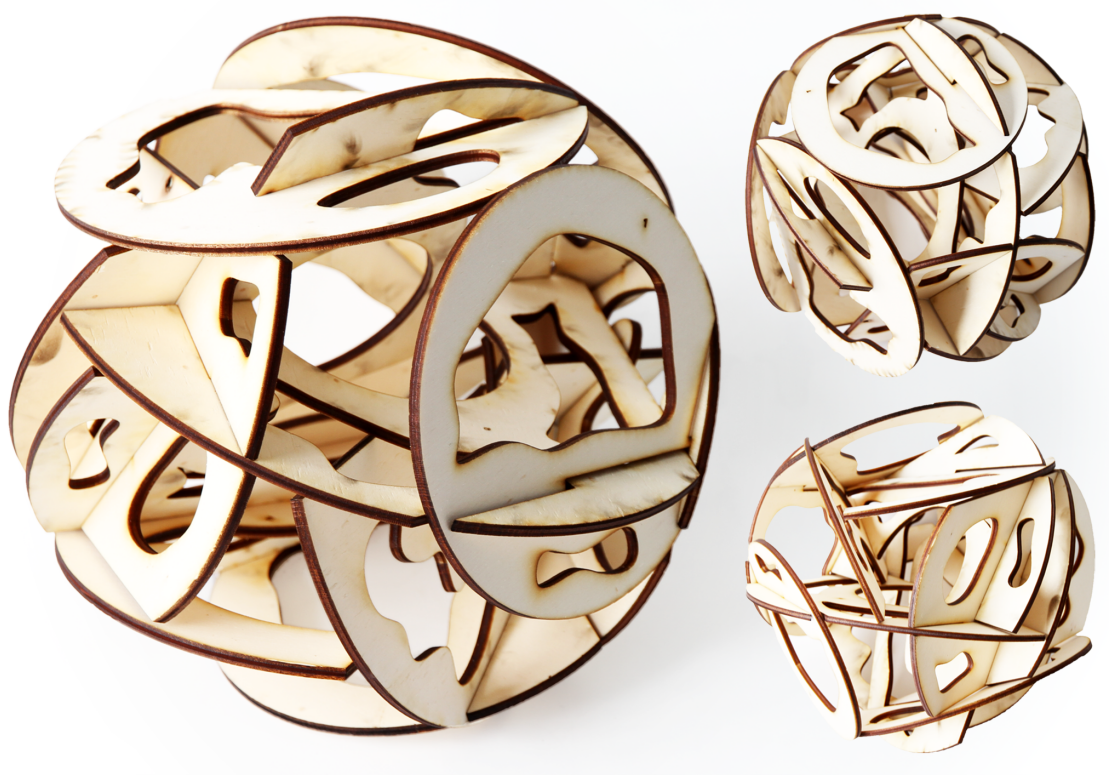


Figure 3.11: A lampshade design uses non-tight slits to facilitate a complex intersection pattern. Our optimization still guarantees that the assembled structure is completely rigid.

trated in Figure 3.10 and the supplementary materials, and the system shows animations of the assembly to aid the user. All our examples have been physically fabricated and assembled, demonstrating that our system covers the full chain from digital design to production.

Limitations. We use a greedy clustering strategy to generate a series of parallel cuts in order to avoid the full combinatorial search that would be computationally intractable for all but the simplest graphs. Hence we are not guaranteed to find a global optimum in the sense of closest valid configuration to a given set of input constraints. In our experiments, however, we found that the greedy choice was as good or better than all subsequent clustering or user-assisted choices for cuts.

Currently, we do not incorporate statics or material physics into the optimization. Similarly, we ignore potential limitations of the production process, such as restrictions of the tool path for milling machines, or specific properties of the material, such as

anisotropies (e.g. in wood) that would favor certain directional alignments. These limitations offer a trade-off between the effectiveness of the design approach and the computational complexity. They are a consequence of the specific abstraction of our model that focuses primarily on the geometric relations and properties resulting from the main fabrication and assembly constraints.

Future Work. The limitations discussed above are one immediate target for future research. Integrating static properties or other performance objectives into the optimization provides numerous opportunities to improve the effectiveness of the design process, perhaps with a method similar to [UIM12]. The process would also be aided with additional semantic controls and shape-aware operations. Another promising avenue for future research is to consider a broader spectrum of material behavior. For example, bendable pieces made of thin wooden or metal sheets can be fabricated with the same technology as the rigid pieces currently considered in our work. Developable surfaces lead to constructions that are much more general, but also more difficult to control.

Further potential for future work lies in the assembly process itself. Our goal previously was to make assembly as easy as possible. Yet some applications might target the opposite. For example, 3D puzzles are intriguing because finding an assembly sequence is difficult and often requires playful experimentation [LFL09, XLF+11]. We believe that interlocking planar pieces have a great potential for challenging puzzles, even when using only few pieces. For example, the 7-piece cyclic model shown in Section 3.6 was perceived by several test persons as very difficult to assemble without explicit assembly instructions. This example illustrates the potential of our approach for creating appealing 3D puzzles.

3.6 Orthogonal Intersection

We provide a brief analysis of the design space for orthogonally intersecting planar pieces with tight slits. The angle constraint for strong rigidity effectively eliminates a degree of freedom in the relative orientation of connected pieces. Only a rotation around the respective plane normals is possible. Consequently, any two non-parallel planar pieces $\mathbf{p}_i$ and $\mathbf{p}_k$ can be connected by a piece $\mathbf{p}_j$ with unique plane normal $\mathbf{n}_j = \mathbf{n}_i \times \mathbf{n}_k$. In the special case where $\mathbf{p}_i$ and $\mathbf{p}_k$ are parallel, i.e. $\mathbf{n}_i = \mathbf{n}_k$, any plane with $\mathbf{n}_j \cdot \mathbf{n}_i = 0$ will be a valid connection. These restrictions on the connection of pieces have interesting consequences on the design space.

Conceptually, assuming consistent slit orientations, a set of parallel planes connected to the same pieces can be collapsed to a single piece. An example is given in Figure 3.12, where the constraint graph is a bipartite graph. The vertex set shown on the right can be collapsed into a single vertex, which eliminates all cycles and thus trivially ensures that the graph can be assembled. Densely connected graphs of this sort naturally lead to parallel planes in the arrangement. In this particular example, all nodes have valence $n/2$, where $n = |\mathcal{V}|$. As a result, any cut has at least $n/2$ edges. To ensure that the graph can be split along the cut, these edges need to be parallel. This leads to at least $n/2$ parallel planar pieces.

This example suggests that parallel edges of a cut lead to parallel planar pieces. However, this is not necessarily the case. Let us consider a simple elementary cycle consisting of n pieces, i.e. $2n$ unknown parameters for the n plane normals (see Figure 3.13). The available degrees of freedom are reduced by angle constraints, the global rigid motion of the structure, the constraint that the pieces form a cycle, and the existence of a parallel cut, which necessitates that (at least) two edge directions must be parallel. The formula below quantifies the degrees of freedom $F(n)$ of an elementary cycle of size n:

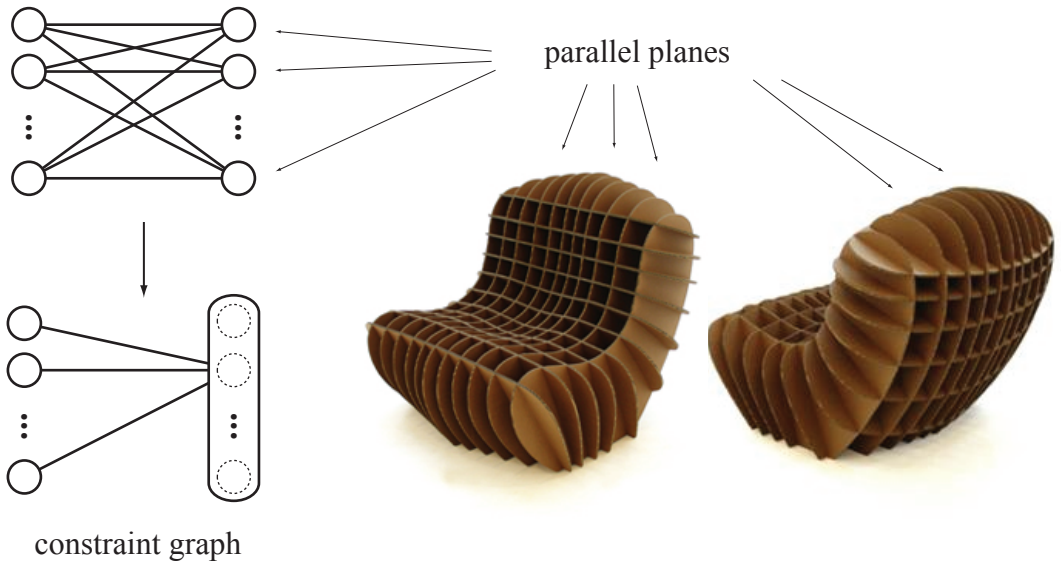


Figure 3.12: A typical grid design commonly observed in existing cardboard models requires one family of planes to be parallel. These types of constructions are often called *waffle grids*. In the constraint graph, these parallel planes can be collapsed to a single node, leading to a configuration without cycles that trivially ensures that all slit constraints can be satisfied.

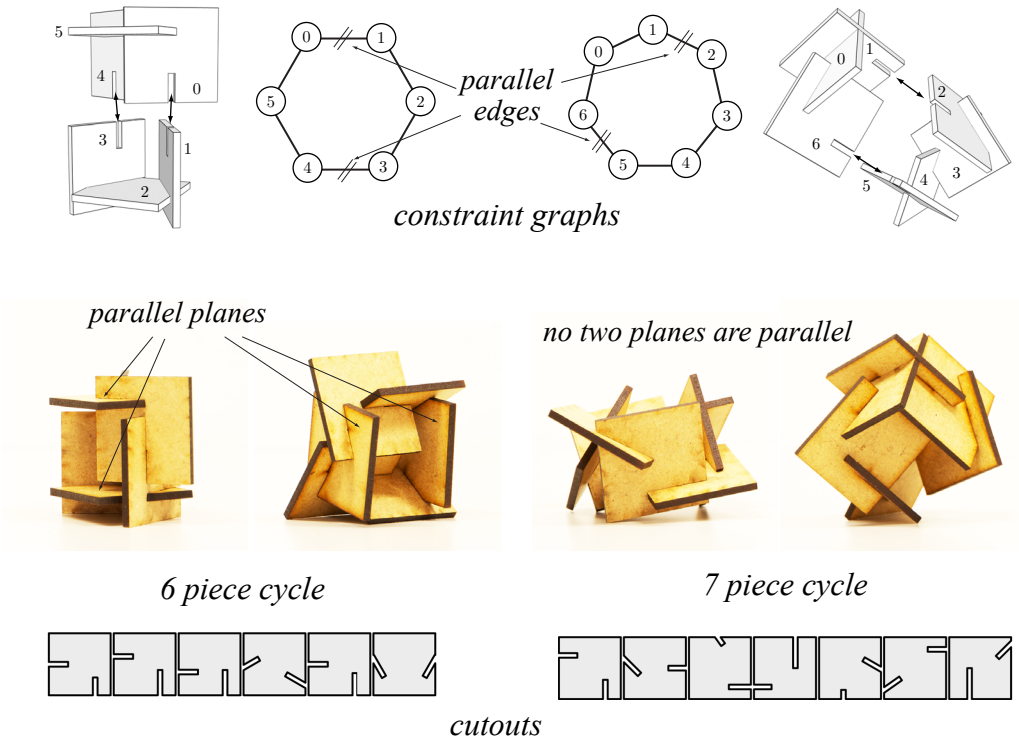
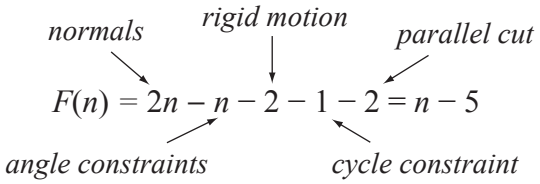


Figure 3.13: Elementary cycles with orthogonally intersecting planes. A parallel cut consisting of two edges ensures that the cycles can be assembled. With six or less pieces, such cycles must contain at least two parallel planes. While not specifically designed as puzzles, the assembly of these models can be challenging without a given assembly plan, since the parallel cut is the only way to close the cycle. This illustrates the potential of our approach for generating recreational 3D puzzles.



Consequently, for a cycle with 6 pieces, the single remaining degree of freedom is the rotation of the two pieces associated with a parallel edge around the axis defined by the edge direction. As it turns out though, in this case the two pieces connecting the parallel edges must be parallel. Thus the smallest cycle that can be strongly rigid and assemblable and does not contain any parallel pieces consists of 7 pieces.

3.7 Design Process Details

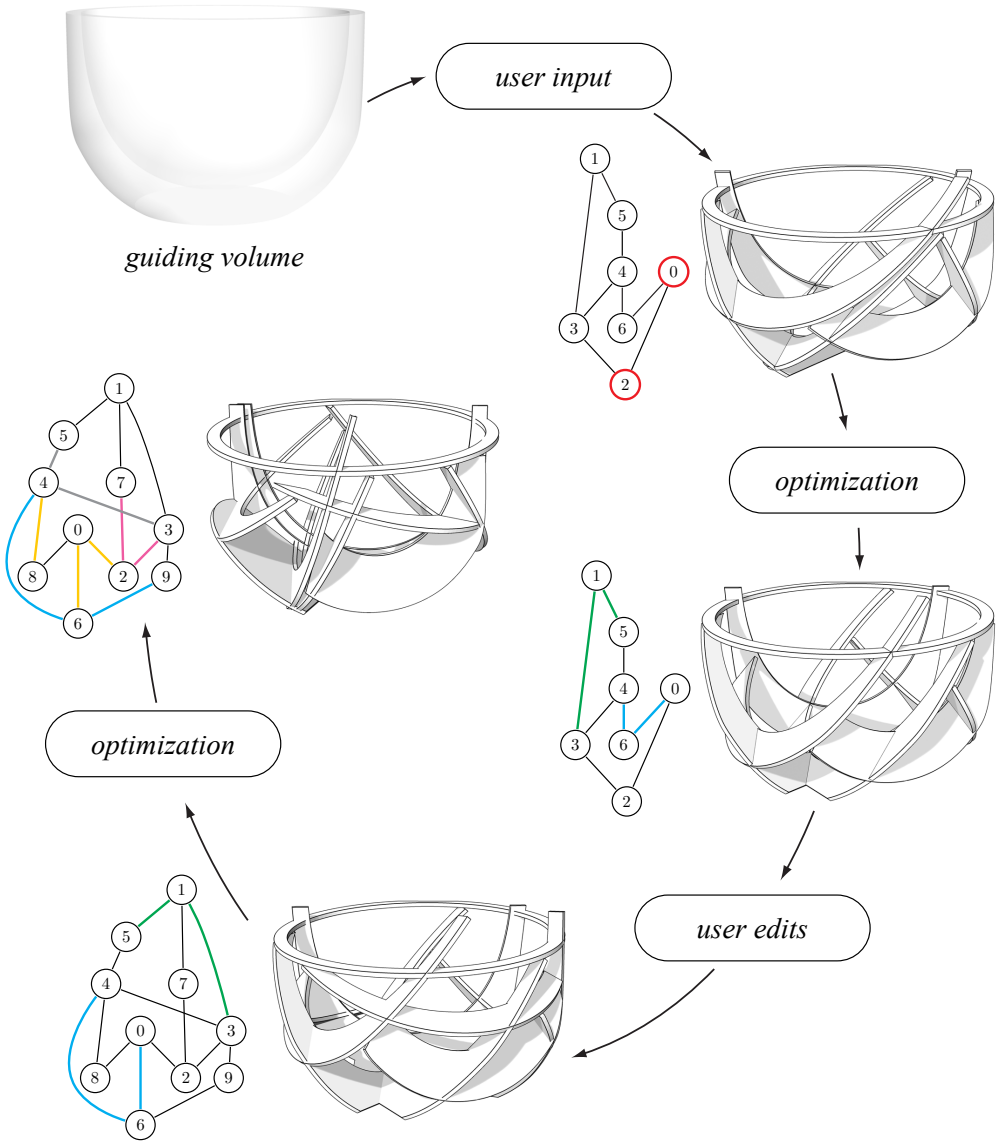


Figure 3.14: Illustration of a typical design process. The user first specifies a 3D guiding volume, then positions and orients several initial pieces. The constraint graph automatically computed from the intersections typically violates some angle constraints (red nodes) and/or does not contain the required parallel cuts. The optimization then solves for a configuration that satisfy the constraints (colored edges are parallel). The user adds more planar pieces and iteratively refines the design, relying on the optimization method to ensure constraint satisfaction.

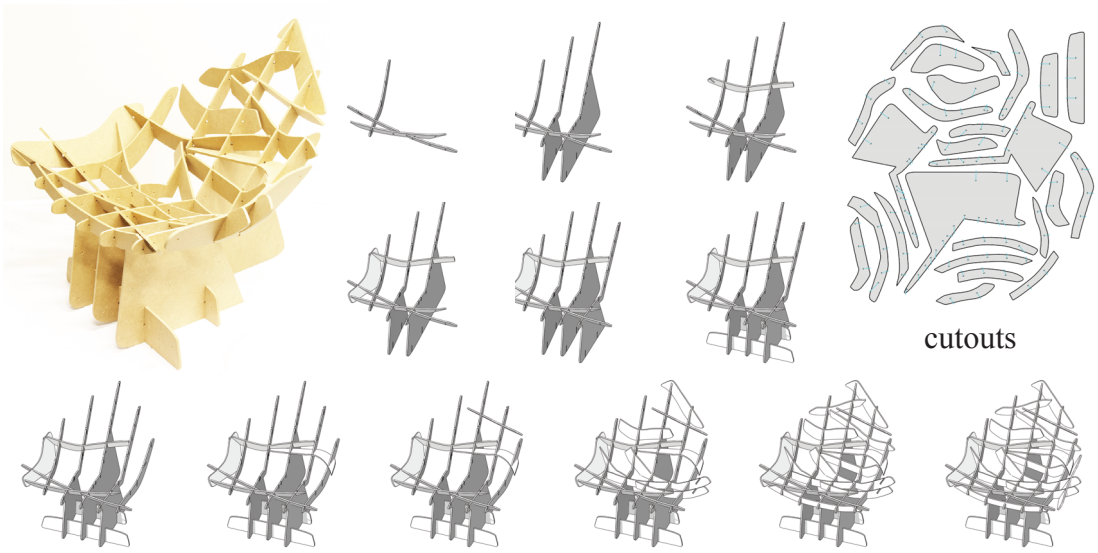


Figure 3.15: Assembly sequence of the freeform chair model.

Figure 3.14 illustrates a typical design session. We deliberately chose an example where the algorithm leads to significant change of the plane orientations to better visualize the effect of the optimization. While these alterations are often more subtle, they can involve a large subset of pieces. Satisfying the constraints by manually moving and rotating the pieces would thus be very cumbersome, even for simple designs consisting of few pieces. At the end of the process, assembly instructions can be directly generated from the assembly graph, see Figure 3.15.

We present a comparison of plane orientations and positions before and after a single step of optimization in Figure 3.16. For inputs we use sample meshes and planes generated by the method of McCrae et al. [MSM11].

Implementation. Our framework is implemented in C++ using the Qt library. We use the Boost Graph Library for all graph operations and Boost Geometry for performing CSG operations and collision detection. We also use the C Clustering Library written by Michiel Jan Laurens de Hoon for clustering the orientations. We use Sequential Least-Squares Quadratic Programming as implemented by NLopt [Kra94, Joh10] to solve Equation 5.

In very large graphs (> 500 edges), decreasing the number of clusters by only one per iteration of the clustering algorithm can be slow, therefore we trade accuracy for speed and decrease the clusters by a small multiple of the number of edges in the cycle. Sim-

ilarly, for large graphs, discarding only one constraint per iteration of SQP can cause many iterations, and we discard a small number of edges at once, again a trade-off between accuracy and speed.

The SQP solver for Equation 5 requires as input the gradient of each constraint, which can be assembled from the partial derivatives:

$$\frac{\partial c_{\text{angle}}}{\partial \mathbf{n}_i} = 2\left(\mathbf{n}_i \cdot \mathbf{n}_j\right)\mathbf{n}_i$$
$$\mathbf{g}(\mathbf{n}_i, \mathbf{n}_j, \mathbf{n}_k, \mathbf{s}) = \mathbf{n}_i(\mathbf{n}_k \cdot \mathbf{n}_j) - \mathbf{n}_j(\mathbf{n}_k \cdot \mathbf{n}_i)$$
$$\frac{\partial c_{\text{slit}}}{\partial \mathbf{n}_i} = 2\mathbf{g}\left(\mathbf{s} \cdot \mathbf{n}_j\right) - \mathbf{s}\left(\mathbf{n}_j \cdot \mathbf{g}\right)$$
$$\frac{\partial c_{\text{slit}}}{\partial \mathbf{n}_j} = 2\mathbf{s}\left(\mathbf{n}_i \cdot \mathbf{g}\right) - \mathbf{g}\left(\mathbf{s} \cdot \mathbf{n}_i\right)$$
$$\frac{\partial c_{\text{slit}}}{\partial \mathbf{s}} = 2\left(\mathbf{g} \cdot \mathbf{n}_i\right)\mathbf{n}_j - \left(\mathbf{g} \cdot \mathbf{n}_j\right)\mathbf{n}_i$$

The partial derivative of c_{slit} is derived by rewriting the inside of c_{slit} as a dot product expression $\mathbf{g}$.

Performative constraints We can also add performative constraints to our system. As an example of a performative constraint, we consider pieces forming a set silhouette (see Figure 3.17) when viewed or lit from a certain direction, as in Shadow Art [MP09]. This soft constraint can be used for an application such as daylighting or can enable the designer to intuitively fill a section of a certain volume for semantic purposes. The constraint can also be prescribed to necessitate a certain shape or "skeleton" in order to maintain structural integrity. Consider two intersecting pieces A and B. The silhouette constraint is satisfied locally within the solver by for each target silhouette, translating the centroids of each pair of intersecting pieces in opposite directions on the projected plane of the silhouette such that they maximally fill the space in 2D defined by the target silhouette.

Another example, for performative reasons (for example making a flat top and bottom for stability for a table) it is possible to slide planes in the direction of the normal of a piece. These pieces are then held by gravity (see Figure 3.18)

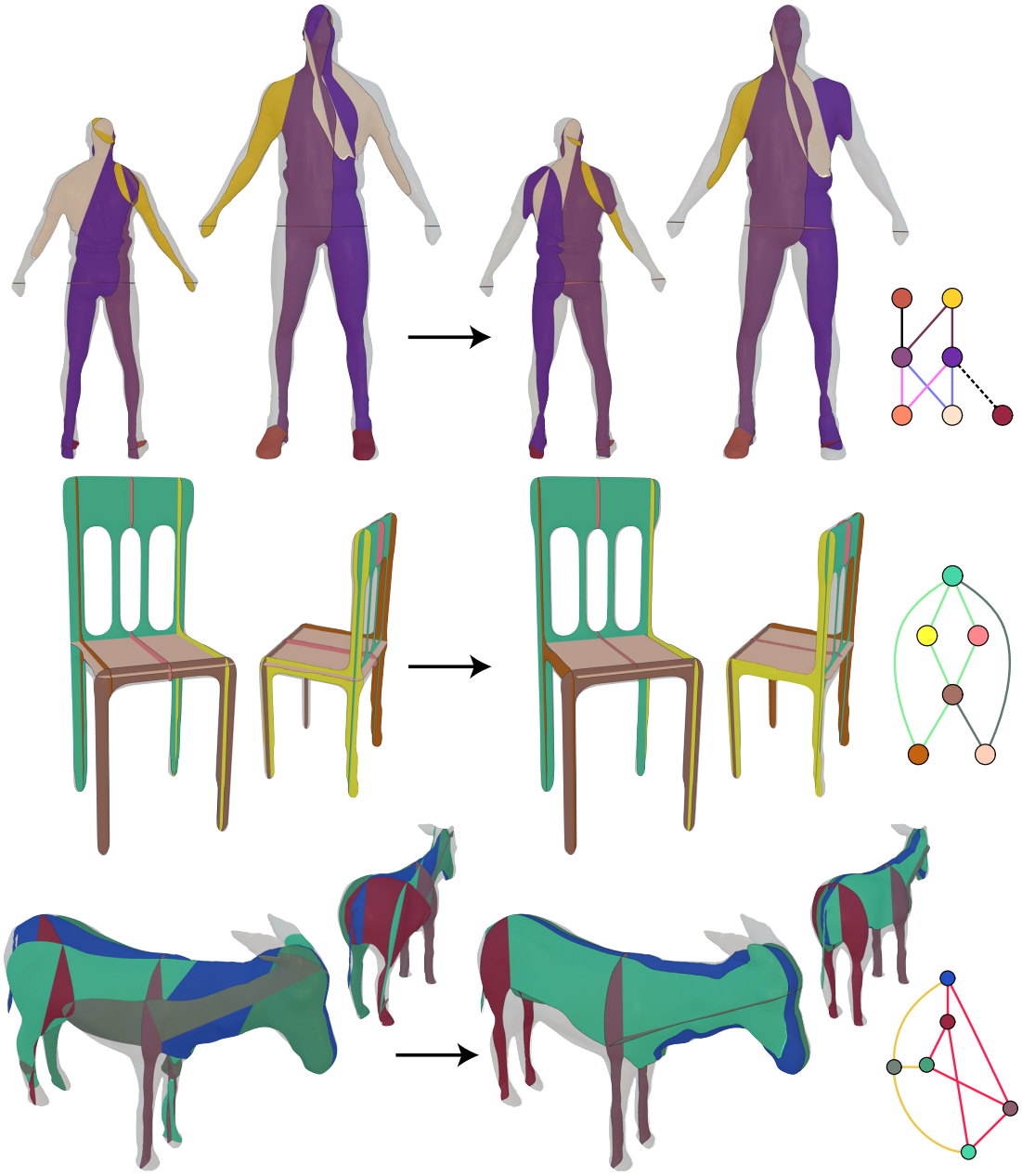


Figure 3.16: We use the method of McCrae et al. [MSM11] to generate initial plane positions (left). After a single optimization step, the results at right are produced. The respective constraint graphs are shown at the far right. In a normal editing session, the user would continue adding planes and constraints from this point.

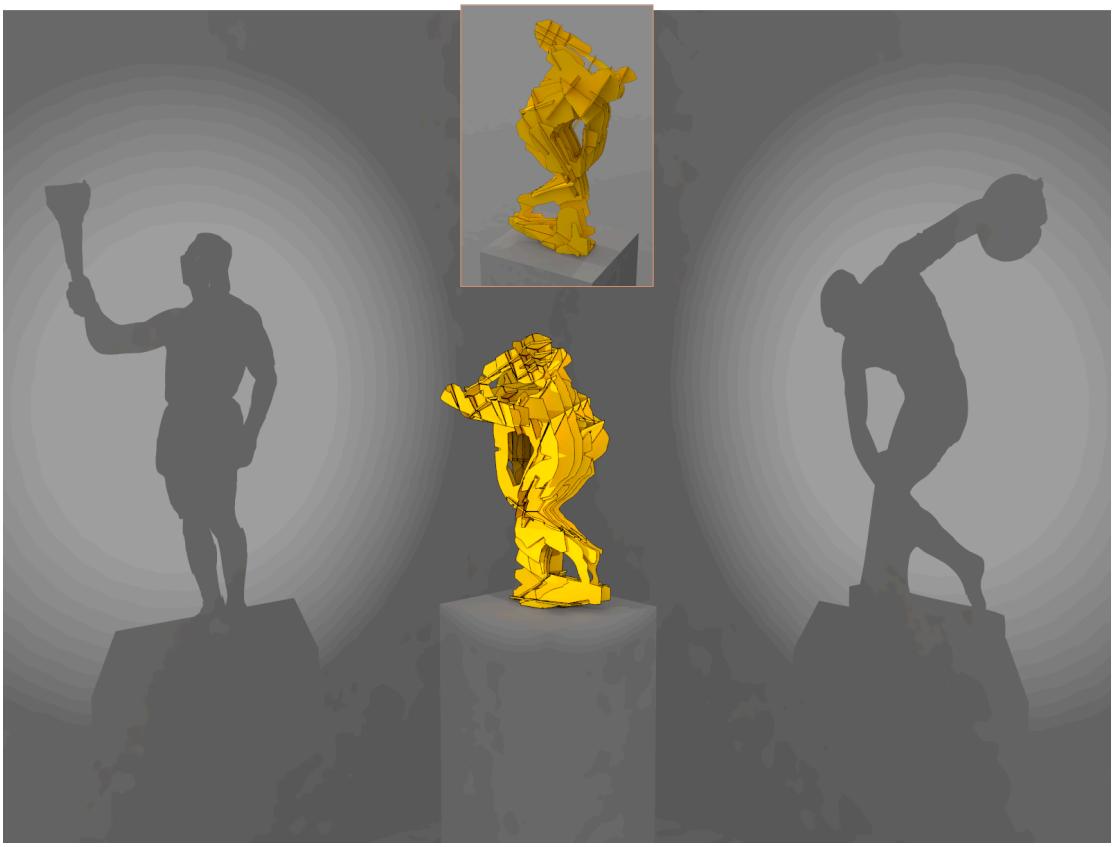


Figure 3.17: A Shadow Art example (see [MP09]) with many more cycles. The example is constructed with silhouette constraints (the two shadows shown). In this case, the optimization is done as a post-rationalization step, and as there are many cycles, pieces tend to become more parallel.

3.8 Additions and remarks

Extending contouring In the system, a user specifies a boundary using a planar section of a guiding volume (see Figure 3.14) of an existing 3D object. After adding many planes, this leads to an overly-constrained system where the result is very likely to be a grid design (as in Figure 3.12). The system as proposed solved this problem by allowing the user to cut planar slices before intersecting other slices. These boundaries can be modified by the artist in a post-process and are guaranteed to still satisfy the constraints as long as the artist does not expand the boundary or remove too much from the slit regions. However, this is often a tedious process that leads to undesirable outcomes, particularly if the goal is simply to illustrate a 3D shape and the internal structure is not important.

Figure 3.18: A small table built with orthogonally intersecting planar pieces using our design method. Also pictured are the hull wireframe, the negatives of the corresponding pieces, and the assembly process. This example features planes assembled in directions parallel to their normals, which can be held simply due to gravity.

Cignoni et al. [CPMS14] propose a method using ribbon-shaped slices to reduce the constraints. These ribbons follow an input cross-field specified on the surface, following the method of Bommes et al. [BZK09]. The cross-field can also be optimized for

symmetry using [PLPZ12]. The advantage of these ribbons is that the general look of the 3D shape is preserved while the number of constraints is kept low. This comes at the cost of not having the rigidity of a full internal structure.

However, this fully automatic process does not involve the user besides allowing for positional and direction constraints in the cross-field. Another limitation of their method is that assembly is constrained to be moved as one piece at a time rather than allowing for the full degree of freedom of cuts across multiple pieces. Therefore, a combination of the two methods would be best for a design process that leaves as many degrees of freedom as possible to the user. This will be discussed in the following pages.

In our method, during contouring, when there are ambiguities we split the piece with the most intersections in order to maintain the greatest amount of connection in the graph (see Section 3.3.2). We can use another idea introduced by Cignoni et al. instead. We can choose the intersection to maximize the isoperimetric number $h(G')$ of the dual graph [BHT00], where $G' = \{V', E'\}$ and V' denotes the intersections while E' denotes the pieces. The isoperimetric number is

$$h(G') = \min \frac{|\delta A|}{|A|}, \tag{3.8}$$

minimized over all subsets $A \subseteq V'$ where the size of A is $0 < |A| \leq \frac{|V'|}{2}$. δA is the *edge boundary* of E', the set of edges with exactly one endpoint in U. If this number is small then there exists two large sets of vertices with few edges between them. If it is large than any possible division of V' into two subsets has many edges between those subsets. Choosing based on this criterion favors a well-connected graph.

Relaxing the constraints Cignoni et al. [CPMS14] also further relax the slit constraints, allowing non-parallel slit directions in the assembly process. They observe that the slit intersections can be relaxed while preserving rigidity. This comes from the fact that there are multiple ways to ensure rigidity from non-orthogonal intersection angles. One is a triangular arrangement such as Figure 3.3(f). Another is four planar pieces interlocked with non-parallel slit directions—this intuition is due to the non-orthogonal slit acting as a hinge and the four connected pieces forming a four-bar linkage [MS10]. For stability, at least one of the intersections in this linkage must be tight, and therefore an angle constraint must be included on this to keep it rigid.

If this relaxation is included in our system, there are more degrees of freedom and therefore the user input can be matched more closely. This can then be used to relax the hard constraint c_{slit} in Equation 3.5. This can be reformulated as either a soft constraint

or a change in the algorithm when Equation 3.5 does not yield a solution.

If Equation 3.5 has no valid solution, we remove the angle constraint on an edge with the highest c_{angle}. Under this relaxation, we can also remove the slit constraint on this edge.

However, alternatively, we can reformulate c_{slit} and c_{angle} as soft constraints and minimize instead the following optimization. Due to the relaxation of the slit constraints, as well as for better rigidity, it is important to keep the slit width from getting too wide. Therefore, we try to minimize each term instead of just dropping it.

$$\arg\min_{\mathbf{n}_1,\ldots,\mathbf{n}_N} E_{\text{input}} + \omega_1 \sum_{\mathbf{e}_{ij}} c_{\text{angle}} + \omega_2 \sum_{\mathscr{S}_k} \sum_{\mathbf{e}_{ij}\in\mathscr{S}_k} c_{\text{slit}}$$

$$\begin{aligned} \text{subject to } \|\mathbf{n}_i\|^2 &= 1 \ \ \forall \mathbf{n}_i, \\ \|\mathbf{s}_i\|^2 &= 1 \ \ \forall \mathbf{s}_i. \end{aligned} \tag{3.9}$$

where ω_1, ω_2 are user-defined weights. Due to the formulation of Equation 3.9, we can use an unconstrained minimization and parametrize n_i and s_i as unit length vectors. For this, we use the Ceres optimization framework [AMO13]. Alternatively, we can reproject $\mathbf{n}_i$ and $\mathbf{s}_i$ after each optimization step. Reformulating this optimization in an unconstrained manner allows speed up of the algorithm for a much larger number of pieces and makes it unnecessary to reject constraints one at a time. However, there is a trade-off that the error is spread in a least-squares manner over the pieces rather than having the constraints unsatisfied only in certain regions. This can be addressed by solving the problem with an l_p-norm where $p \leq 1$.

Note that as we have a degree of freedom, and to make the cycle stable, we do not use the final $\mathbf{s}_i$ as the slit directions but rather we choose the slit direction $\mathbf{n}_i \times \mathbf{n}_j \in \mathscr{S}_k$ that is closest to $\mathbf{s}_i$. This ensures that at least one of the slits locks tightly along the length.

The weights ω_1, ω_2 denote the amount the user wishes to deviate in order to achieve better rigidity. A useful direction of future work is to adjust ω_1, ω_2 locally in order to balance external forces on the structure and ensure stability.

Future work An important future direction of this work is making it accessible to common users. Since this research, Autodesk 123D Make has made an accessible system to fabricate simple models out of laser cut pieces. This software has been used for scaffolding for other fabrication methods such as the Wire Mesh system of Garg et al. [GSFD+14]. However, the resulting panel layout is highly limited since construc-

tion is ensured trivially by using waffle grid and waffle grid-like patterns. McCrae et al. [MUS14] recently introduced an interactive system to design structures out of intersecting planar pieces. They use constraints derived from our work, as well as static constraints, and provide an intuitive interactive design system. Interesting future work would be to combine a similar interactive system with statics with the optimization in a framework with which architects and designers are already familiar such as Rhinoceros.

Finally, Saakes et al. recently proposed an interactive system to pack polygons to be laser cut to efficiently utilize the material [SCMI13]. Our system would be enhanced by including an easier way to layout 2D parts before fabrication.

Chapter 4

Automatic Generation of Constructable Brick Sculptures

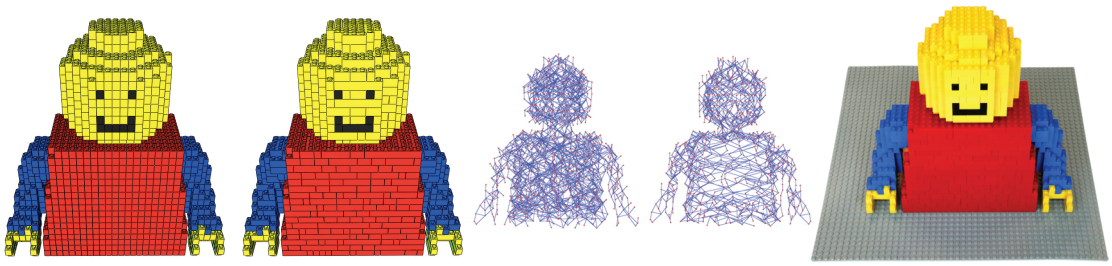


Figure 4.1: A demonstration of our method from start to finish. The LEGOMAN is first voxelized into 1×1 bricks and the bricks are merged respecting color. Then, the bricks are optimized for structure and extraneous bricks are removed. Finally, instructions are produced and a LEGO model is built from the instructions.

The previous chapter proposed an example of a system where optimization works in parallel with artist decisions. However, for certain types of fabrication technologies, an automatic system is more desirable. One example is to construct 3D sculptures out of existing building blocks where the differences in internal structure do not change the aesthetics of the overall piece. The LEGO® construction system can be found in many homes. It provides a way of prototyping that is accessible to many people that do not have access to CNC machines. Building such models in large scale requires careful preplanning to produce constructable and stable models. We propose a system that, starting with a voxelization of a 3D mesh, merges voxels to form larger bricks, and then analyzes and repairs structural problems, finally outputting a set of building instructions. We also present extensions such as producing hollow models, fulfilling limits on the number of

bricks of each size, and including colors. Results (both real and virtual) and timings show significant improvements over previous work.

4.1 Foreword

The generation and rationalization of 3D models for fabrication has recently become a topic of great interest in the computer graphics community. However, most fabrication methods still require the user to have expensive equipment such as laser cutters or 3D printers. LEGO®, a popular toy construction system, is comparatively cheap and nearly ubiquitous. However, building arbitrary 3D models out of LEGO manually often involves significant trial-and-error. This process requires approximating a 3D model out of a limited set of pieces and ensuring the sculpture to be connected, stable and constructable. The goal of this chapter is to automatically create the set of instructions for a LEGO model from a 3D object representation. By doing so, we highly simplify the task of building a large customized LEGO model.

The LEGO Group, the company that produces LEGO toys, has twice openly presented this problem to the scientific community—first in 1998 and later in 2001 [Pet01]. Our approach is inspired by previous work [vZS08] in the sense that the algorithm starts from the voxelization of a model into the smallest possible bricks, namely the 1×1 bricks (see Figure 4.3) and merges those to form larger bricks. However, merging bricks greedily makes it highly unlikely that the model is buildable or stable. To resolve this, we propose a graph-based algorithm to ensure brick connections and resolve structural weaknesses. Furthermore, our method can reduce the brick number by hollowing the model, allows the user the specify limits on how many bricks of a certain type can be used, and can even take into account the coloring of bricks.

Previous methods are based on variations of the heuristic

$$\begin{aligned} \text{Fitness} := {} & C_{numbricks} \times numbricks + \\ & C_{prepend} \times prepend + \\ & C_{edge} \times edge + \\ & C_{uncovered} \times uncovered + \\ & C_{otherbricks} \times otherbricks + \\ & C_{neighbor} \times neighbor \end{aligned}$$

where the C are weights and for each layer:

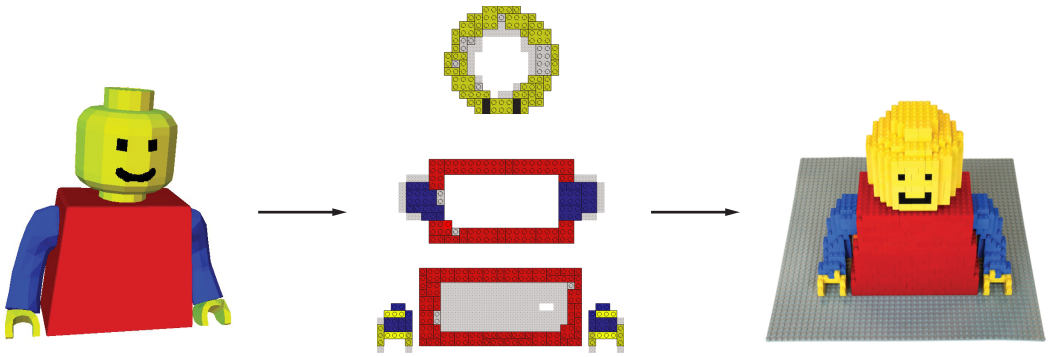


Figure 4.2: The construction process. First, a triangle mesh with color data is given to our method. Our method generates layer-by-layer instructions that can then be followed by the user in order to build the model.

- *numbricks* is the number of LEGO bricks;
- *prepend* is the directionality of bricks in consecutive layers;
- *edge* is the number of edges of each brick that coincide with edges in the layer below;
- *uncovered* is how much area of each brick is not covered by bricks in the neighboring layers;
- *otherbricks* is the number of bricks in the layer below not covered by bricks; and
- *neighbor* is how far from each brick's edge is the edge of a neighboring brick.

This heuristic is difficult to optimize for and the weights are tricky to find. Contrary to this we use a simple metric that corresponds to the stability of the model and can be optimized for efficiently.

4.2 General pipeline

The goal of our approach is as follows: given a 3D model as a triangle mesh and color data, we want to generate layer-by-layer building instructions so that a user can easily assemble the given LEGO model such that it will be connected and stable (see Figure 4.2).

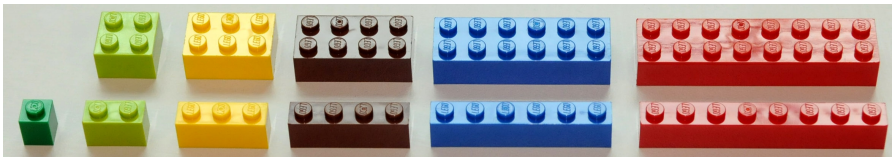


Figure 4.3: The set of legal bricks.

As in previous methods, we first represent the object as 1×1 LEGO bricks. We transform a mesh representation of an object into a discrete set of voxels using the method of [NT03]. The choice of voxel resolution is important as a higher value results in a better approximation of the original object, but would require more bricks and more time to build. We optionally hollow the voxelization, leaving only a given number of outer layers in order to use less bricks to build the same object.

After conversion, we sequentially merge the 1×1 bricks in a greedy fashion. Given a legal set of bricks, we make them as large as possible until no further merges can be done (see Figure 4.3). At this point, the model is very likely to be weakly connected and possibly disconnected leading to an unbuildable structure. Therefore, we increase the solidity of the model by identifying the weaknesses and repairing them.We then

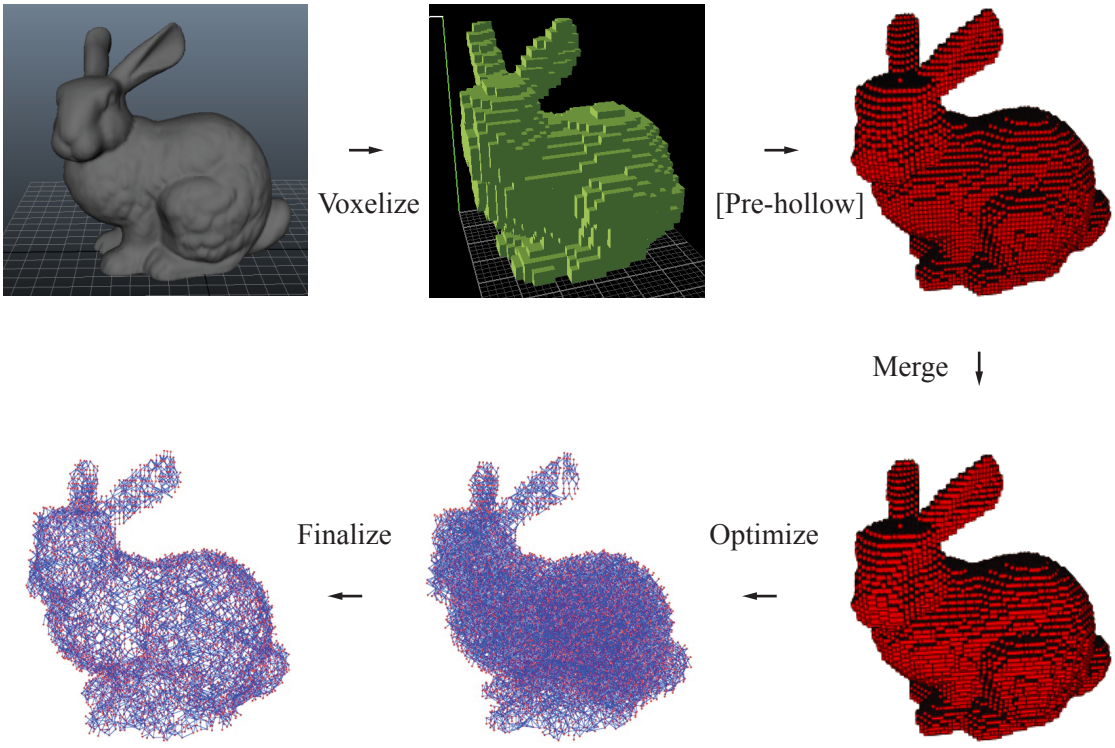


Figure 4.4: The pipeline of our approach for the BUNNY model.

optionally remove bricks if they don't modify the solidity. Finally, the user can save the instructions as images representing the brick layout of each layer to facilitate building, or a video of the building process can be generated. A graphical representation of the process can be seen in Figure 4.4.

4.2.1 Merge algorithm

In order to decrease brick count and increase connectivity, we prefer larger bricks. We can do this simply by merging bricks with their neighbors. We use a randomized greedy merge algorithm as follows:

1. Choose a brick in the model at random.
2. Find the legal set of neighbors with which the brick can be merged.
3. Select the neighbor with the lowest cost value and merge.
4. Goto step 2 until there are no more mergeable neighbors.
5. Goto step 1 until no brick can merge.

Note that for step 2 only a specific set of LEGO bricks is considered (the *legal bricks*). For our examples, we use the set shown in Figure 4.3, but arbitrary other legal sets can be specified by the user. For step 3, we favor the brick which when merged with the current brick, will create the most connections (two bricks are connected if they are on adjacent levels and they have at least one knob overlapping). If two merges create the same number of connections, we choose between them randomly. This allows us to save optimization steps in the next section. The result of this algorithm for a 10×10 grid can be seen on Figure 4.5.

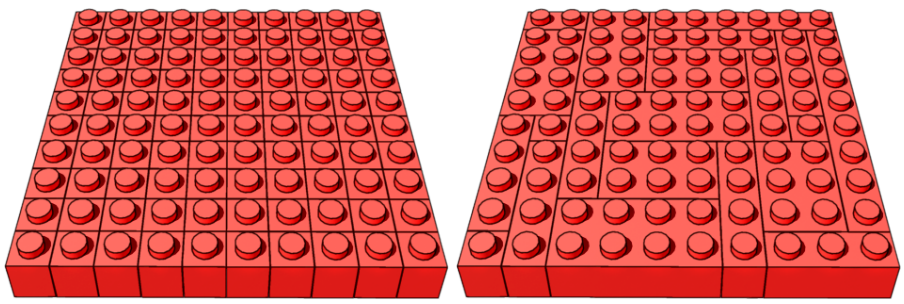


Figure 4.5: The initial merge step.

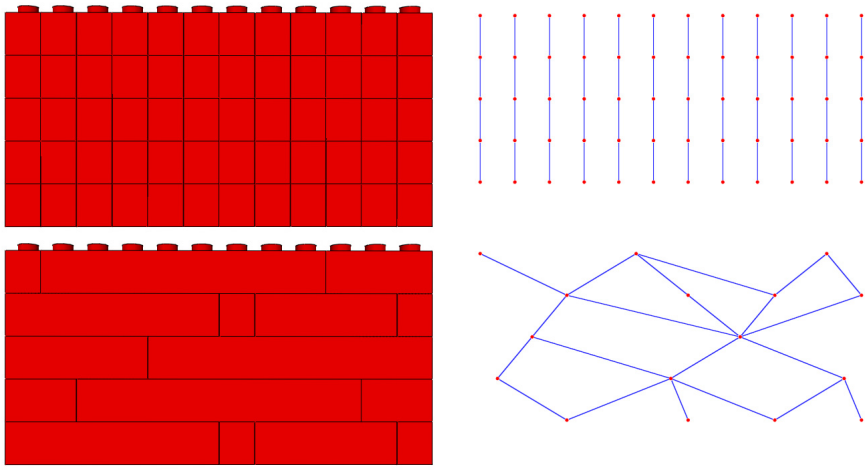


Figure 4.6: Two brick layouts (left), and their respective graph representations (right).

4.2.2 Solidity Optimization

The stability of the construction is related to how the bricks are connected: the more the bricks of a model are connected to each other, the stronger it will be. This observation motivates the mapping of our LEGO brick representation to a graph representation where each brick represents a vertex and each connection between two bricks represents an edge. In Figure 4.6, we illustrate the equivalence between a toy brick layout example and its associated graph; see Figure 4.1 for a more complex example. With this representation, we can analyze the connectivity of the LEGO model to determine weak points.

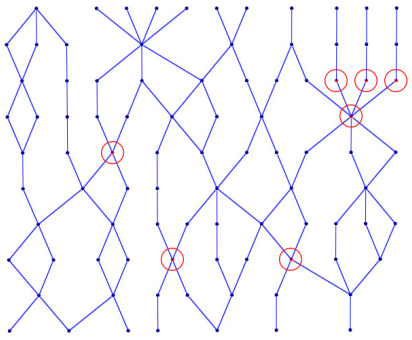


Figure 4.7: A graph with articulation points. Articulation points are shown in red.

The number of connected components in the graph directly relates to solidity as pieces can simply fall off if they are not connected to the rest. Two (non-trivial) subgraphs that are only connected to each other by one brick also weaken the structure. We call the brick connecting subgraphs of size greater than 1 a *weak articulation point*.

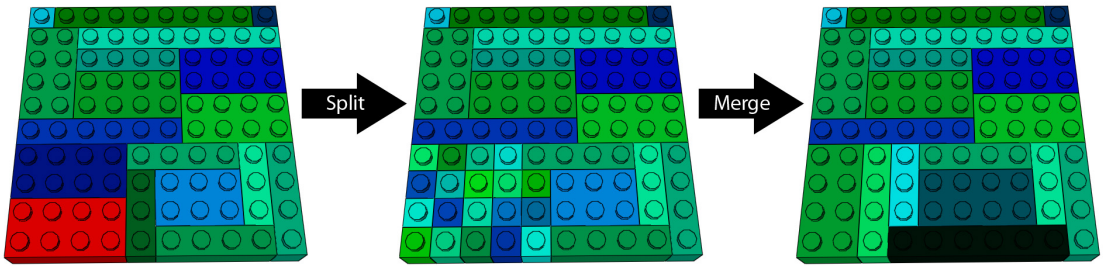


Figure 4.8: The process of removing a weak articulation point.

Using these measures, we change the brick layout to increase solidity. After Section 4.2.1, the bricks are at their maximum extent. Therefore, we split each of the bricks at the interface between two connected components or neighboring an articulation point into 1×1 bricks. Then, we simply run the merge algorithm again, changing only the cost function to a random cost function. We perform this process iteratively until the number of connected components and weak articulation points no longer decreases. In Figure 4.8, we can see the split process for an weak articulation point (in red), note that only the layer containing the weak articulation point is displayed for better visibility.

We tested with a dozen models at various scales, and we find that we need under 50 iterations to have no disconnected components and no weak articulation points for most models. Unfortunately, we cannot know beforehand if the algorithm will converge to the ideal case. There may be thin regions where articulation points cannot be removed (such as the ears of the BUNNY at very coarse voxelizations), or specific voxelizations that result in disconnected components. In these cases, the input mesh would need to be changed to result in a completely stable structure. Nevertheless, if the user does not need a specific voxel resolution, they can iterate with increasing voxel count until a completely stable model is reached. See Table 4.1 and the additional materials for experimental results.

4.2.3 Assembly Instructions

After the previous steps are completed, the user can save the assembly instructions in order to build the model. These correspond to the layout of each layer. To help the user align a new layer above a previous one, we show the layer below shadowed (see Figure 4.9).

4.3 Extensions

Besides the basic pipeline, we introduce several extensions to facilitate the building of the model. In order to reduce the brick number and computation time, the model can be pre-hollowed right after voxelization. After running the pipeline, once the model has no more structural weaknesses, several other steps can be performed to facilitate building of the model. The model can again be hollowed to remove unnecessary bricks, and bricks of certain types can be removed to fulfil a specified quantity of each brick. We can also introduce color into the process.

Reducing the overall brick number Reducing the quantity of bricks can allow for easier and cheaper construction without significantly harming the stability of the construction. If the model is filled with bricks it will require much more bricks than if it is hollow. We have therefore devised two strategies for reducing the brick count by hollowing the model.

Pre-hollowing. Before the pipeline, the user can specify a shell size, and we remove the voxels which are further than than that number of bricks away from the outside of the figure in any direction (not just in that plane). One layer of a hollowed Stanford BUNNY with shell size 2 is shown in Figure 4.10b. An advantage of pre-hollowing is a faster optimization process during the pipeline. The user can change the shell size but usually 2 is sufficient for stability even for complex models.

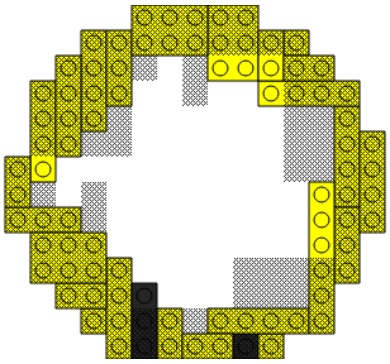
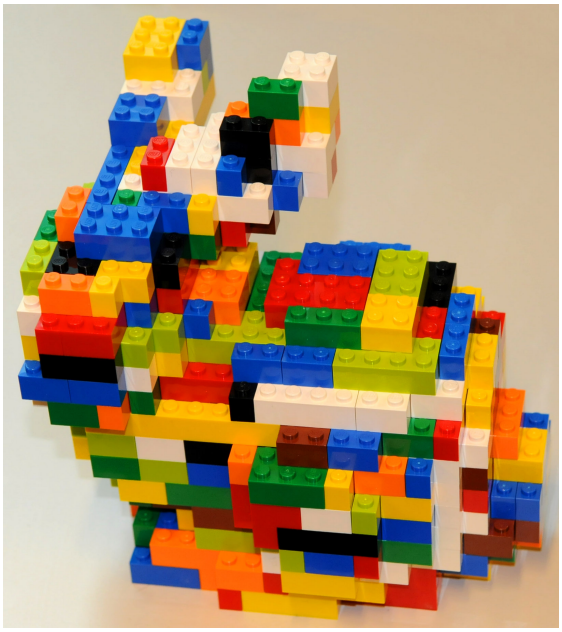
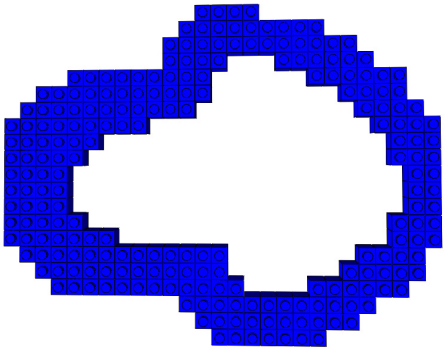


Figure 4.9: Assembly instructions for layer 21 of the LEGOMAN. The two black bricks correspond to the eyes.



(a) The built model.



(b) A pre-hollowed layer before optimization.

Figure 4.10: The Stanford BUNNY model built according to the instructions provided by our method. The voxels are pre-hollowed leading to less pieces and a faster optimization.

Post-hollowing. Pre-hollowing is fast but does not result in a minimal number of bricks. Another technique to reduce brick count is to remove inside bricks without compromising the model solidity. We remove pieces without introducing more connected components or more weak articulation points. For each brick in the inside of the model we consider a subgraph of the connectivity graph centered at the brick. We then remove the brick if its removal does not add any weak points. This method can sometimes result in pillars of bricks that form a path through the middle of the model. Therefore, we can combine pre-hollowing and post-hollowing to start with a shell and then remove any extraneous bricks.

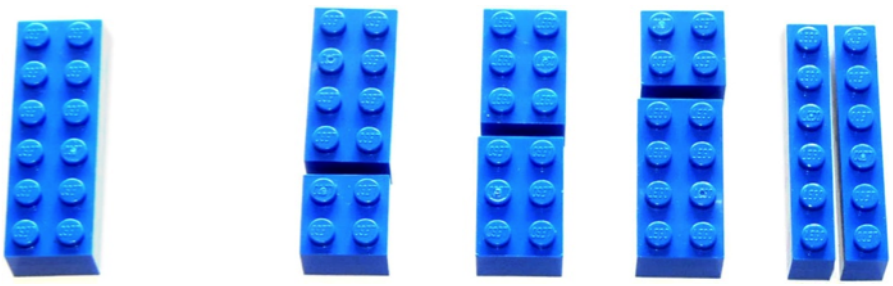


Figure 4.11: The four possible splits of a 2×6 brick.

		no post-hollowing		with post-hollowing		
Mesh	Voxels	Time (sec)	Brick #	Time (sec)	Brick #	Weak Pt
EROS	15144	4.66 ± 1.35	3820 ± 26	5.51 ± 1.40	3110 ± 28	0.10 ± 0.30
BUNNY	11472	26.5 ± 5.98	2900 ± 21	27.0 ± 6.00	2380 ± 30	7.20 ± 2.48
FERTILITY	6859	2.46 ± 1.21	1610 ± 18	2.78 ± 1.20	1400 ± 20	0.05 ± 0.22
KITTEN	12887	2.04 ± 0.73	3340 ± 28	2.72 ± 0.71	2650 ± 27	0 ± 0
LEGOMAN	9961	2.17 ± 0.85	2120 ± 25	2.50 ± 0.84	1800 ± 21	0 ± 0

Table 4.1: Mean values and standard deviations for 20 trial runs of 5 models at 50 layer resolution. Note that the randomized algorithm produces consistent results across different trials. Every trial resulted in a single connected component.

Satisfying brick type limits When building a model with a set kit of bricks, there are set limits to each type of brick. For example, there can be many more 1×2 bricks and not enough 1×4 bricks. Limiting the brick type greedily during merging often does not result in a solid model. We therefore use an approach similar to post-hollowing: As a post-process, we remove bricks over the limit by cutting them into two (legal) smaller bricks. As in post-hollowing, we choose the split as to not to increase the number of connected components or weak articulation points. For example, the possible split operations that can be done on a 2×6 brick is shown in Figure 4.11. If every split causes weak points, we go on to the next brick.

Using colors We can also allow for different colors of LEGO bricks. We initialize the colors for each brick by finding the color of the original mesh texture on the point closest to the center of the brick. We then round this color to the closest LEGO brick color. Then, during step 2 of the merge algorithm (see Section 4.2.1), another verification is added to allow merging of two bricks: if both bricks are outer (visible) bricks and they have different colors, then they cannot be merged. Inner bricks can be merged regardless.

4.4 Results

Table 4.1 summarizes timings and final brick counts for different models pre-hollowed with a shell size of 2 (measurements were done using a 1.8 GHz processor). The time is measured from the start of the first merge to the final result which consists of a single connected component and no weak articulation points. The amount of bricks removed by post-hollowing in each case is close to 20% of the number of bricks before the oper-

ation.

It is difficult to compare the results with those of [vZS08] since they do not have a solidity measure. For example, if we compare the results for the cube with 32 layers, we know that our cube is solid but we cannot say the same for theirs. They report a time of 197 seconds and a brick count of 2,128. If we suppose that both are solid than our algorithm is orders of magnitude faster while using only slightly more bricks.

Using the instructions produced by our method, we built a 17 layer hollow Stanford BUNNY with 314 bricks (see Figure 4.10). We also built a bust of a LEGO figurine (see Figure 4.1) which takes color and brick type limits into account, consisting of 30 layers and using 1,315 bricks.

In Figure 4.14, we evaluate how the number of connected components and the number of weak articulation points evolve over time for three example models. We can see in both cases the number gets sharply reduced during the first few iterations. For the weak articulation points we usually the convergence rate is slower than for connected components. We also show additional results in Figures 4.12 and 4.13. For statistics, see Table 4.2.

4.5 Additions and remarks

The voxelization algorithm may sometimes result in aliasing artifacts. Furthermore, we use the texture of the closest point on the mesh to determine the color of the brick, which can also cause aliasing. These can be replaced with a more sophisticated scheme based on feature detection or by a user-assisted painting and voxel insertion and deletion UI.

The current layer-by-layer instructions have the drawback of making it difficult to add bricks only supported by the layer above. Another way of displaying the instructions would be an interactive visualizer which displays each steps of the assembly similar to the Autodesk Inventor Publisher Mobile Viewer or the LEGO Digital Designer applications. Finally, the code could be parallelized to take advantage of multi-core processors.

Recently, Mueller et al. [MMG+14] have used this algorithm for constructing parts of 3D models out of LEGO used this algorithm for combining 3D printing and our method, so that large parts of 3D models can be built using LEGO in order to save costs and material. Rapid prototyping using existing building blocks would be of great benefit to both professional architects and hobbyists.

Also, by taking into account the weight of each pieces and the gravity, it should be possible to check that the model is stable. If the center of mass is not properly placed to stand in a desired placement, it could be moved by adding pieces in the inside of the model similar to [PWLSH13]. The definition of weak articulation points can be expanded to those that support a load over a certain threshold. This can be further explored by analyzing the stationary and rotational statics of each piece.

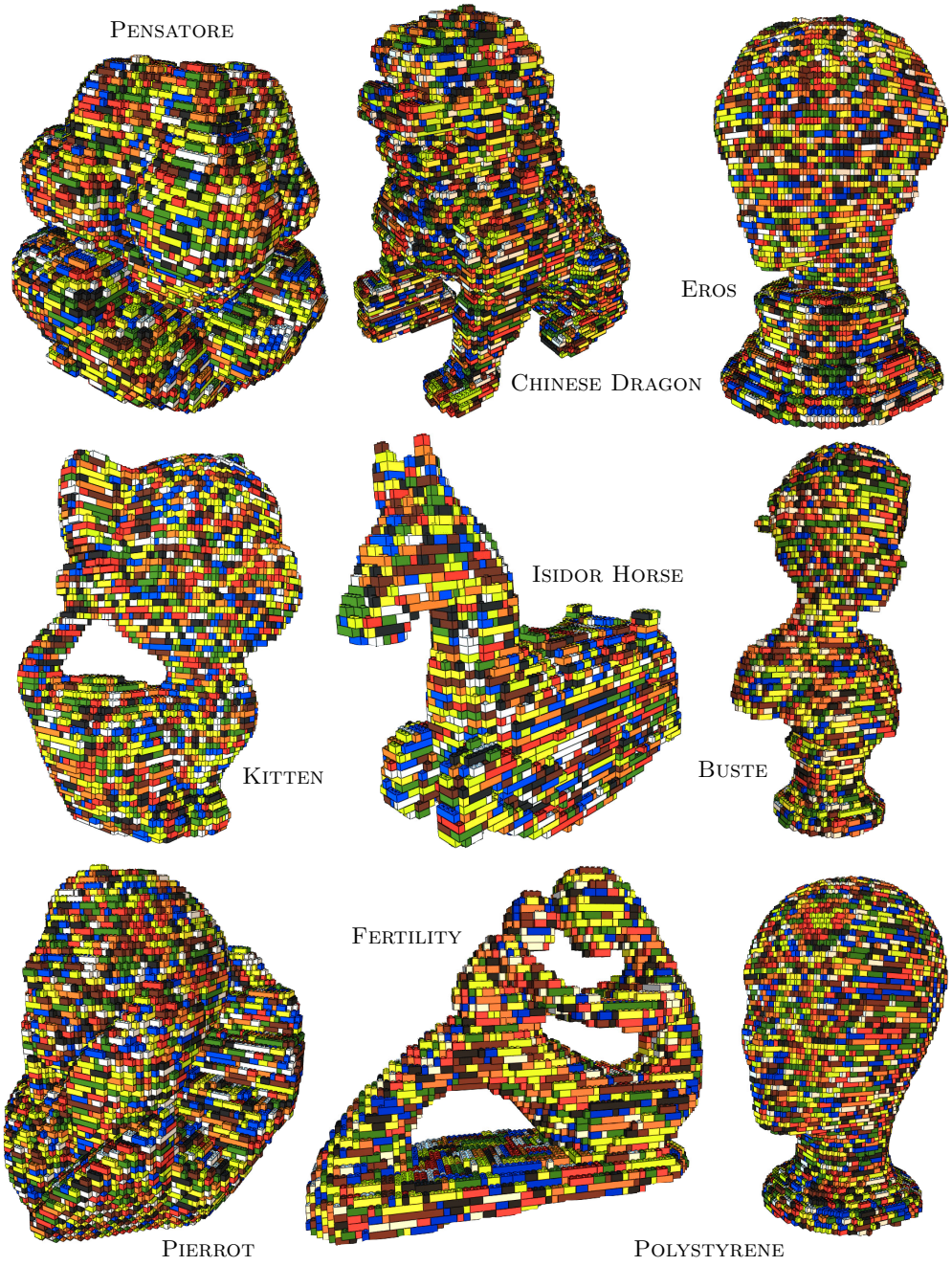


Figure 4.12: Results for several meshes. Colors are random to illustrate each separate piece. Additional information and timings can be seen in Table 4.2.

			no post-hollowing		with post-hollowing				Hollowing
Mesh	Res	Voxels	Time	# Brick	Time	# Brick	# CC	# WAP	Reduction
BUNNY	30	3431	7.77	874	7.89	736	2	12	15.79%
BUNNY	50	11472	12.6	2891	13.2	2383	1	8	17.57%
BUNNY	70	24400	9.47	6068	10.5	4878	1	0	19.61%
BUNNY	90	42604	23.3	10677	25.9	8475	1	0	20.62%
BUSTE	30	2245	1.21	557	1.3	493	2	0	11.49%
BUSTE	50	8026	6.33	2088	6.71	1706	2	2	18.30%
BUSTE	70	17440	9.26	4497	10.2	3577	2	0	20.46%
BUSTE	90	30577	14.6	7872	16.4	6241	1	0	20.72%
CHINESEDRAGON	30	4925	2.56	1262	2.8	1070	3	0	15.21%
CHINESEDRAGON	50	16864	4.46	4209	5.4	3411	1	0	18.96%
CHINESEDRAGON	70	36466	17.6	9009	20	7243	1	0	19.60%
CHINESEDRAGON	90	63631	70.5	15765	74.4	12486	1	0	20.80%
FERTILITY	30	1577	5.92	383	5.94	365	4	8	4.70%
FERTILITY	50	6859	4.21	1556	4.43	1357	1	0	12.79%
FERTILITY	70	16227	2.76	3866	3.59	3241	1	0	16.17%
FERTILITY	90	29210	7.1	6853	8.72	5595	1	0	18.36%
KITTEN	30	3879	0.268	984	0.42	813	1	0	17.38%
KITTEN	50	12887	1.74	3349	2.41	2665	1	0	20.42%
KITTEN	70	26972	8.05	6987	9.51	5525	1	0	20.92%
KITTEN	90	46441	23.8	11915	26.8	9334	1	0	21.66%
LEGOMAN	30	2989	1.47	655	1.55	568	1	0	13.28%
LEGOMAN	50	9961	2.36	2092	2.68	1766	1	0	15.58%
LEGOMAN	70	21362	4.97	4460	5.6	3740	1	0	16.14%
LEGOMAN	90	37478	10.9	7940	12.4	6562	1	0	17.36%
MARIO	30	2263	0.291	573	0.369	515	1	0	10.12%
MARIO	50	8551	8.46	2227	8.82	1896	1	0	14.86%
MARIO	70	19181	13	4868	14.1	3936	2	0	19.15%
MARIO	90	33822	50.4	8669	52.1	6844	1	0	21.05%
PIERROT	30	5877	0.481	1514	0.763	1294	1	0	14.53%
PIERROT	50	18552	5.71	4700	6.67	3759	1	0	20.02%
PIERROT	70	38239	16	9632	18	7635	1	0	20.73%
PIERROT	90	64732	28.4	16426	32.1	13003	1	0	20.84%
ISIDOREHORSE	30	1804	0.91	407	0.961	361	1	6	11.30%
ISIDOREHORSE	50	6680	2.49	1493	2.7	1266	1	3	15.20%
ISIDOREHORSE	70	14692	1.52	3340	2.09	2718	1	0	18.62%
ISIDOREHORSE	90	26022	5.99	5798	6.97	4710	1	0	18.77%
POLYSTYRENE	30	4560	1.27	1214	1.47	973	1	0	19.85%
POLYSTYRENE	50	14615	3.88	3782	4.55	3030	1	0	19.88%
POLYSTYRENE	70	30538	10.7	7890	12.4	6214	1	0	21.24%
POLYSTYRENE	90	51748	18.3	13210	21.1	10405	1	0	21.23%
PENSATORE	30	5898	0.809	1463	1.13	1216	1	0	16.88%
PENSATORE	50	18559	5.02	4743	6.04	3740	1	0	21.15%
PENSATORE	70	38387	12.3	9752	15.1	7450	1	0	23.61%
PENSATORE	90	65290	35.5	16574	39.9	12827	1	0	22.61%
EROS	30	4597	7.81	1155	7.94	1029	2	0	10.91%
EROS	50	15144	8.34	3867	9.28	3140	1	0	18.80%
EROS	70	32208	10.4	8028	12.5	6424	1	0	19.98%
EROS	90	55235	27.1	13764	30.8	11005	1	0	20.05%
								Average Reduction	17.54%

Table 4.2: Additional timings and results for the models in Figure 4.12. The maximum number of iterations was set to 50 for all models. Res refers to the number of layers. Time is reported in seconds. # CC is the number of connected components and # WAP is the number of weak articulation points.

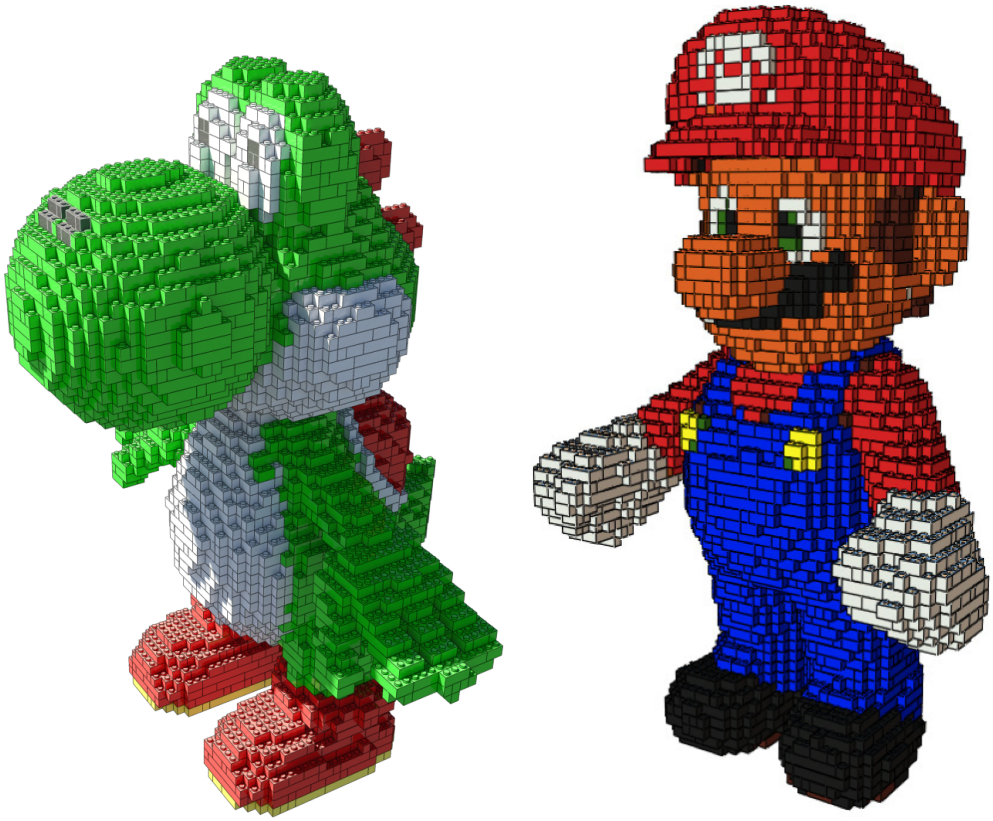


Figure 4.13: Additional results with texture.

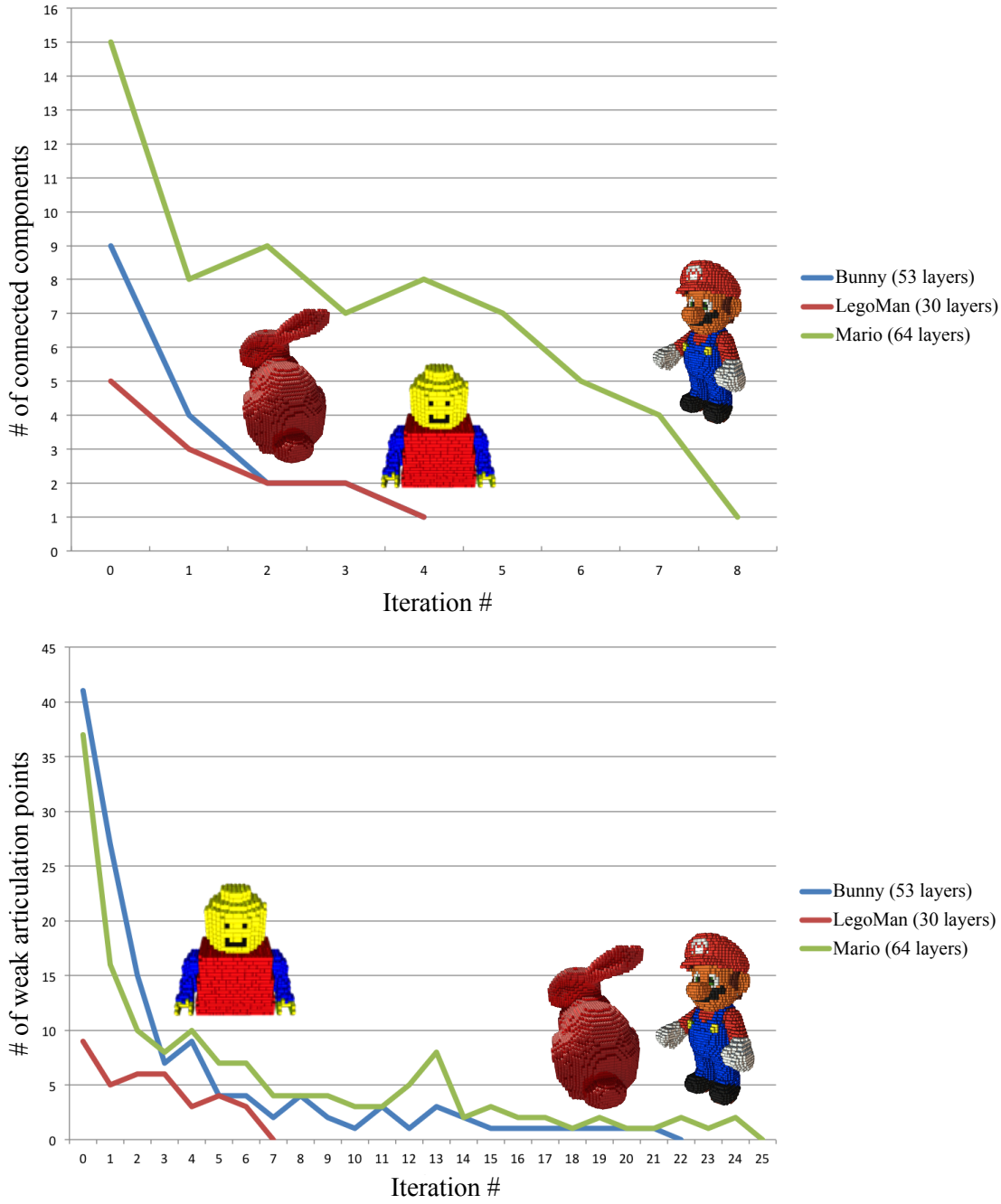


Figure 4.14: The number of connected components and the number of weak articulation points for each iteration for three models.

Chapter 5

High-contrast Computational Caustic Design

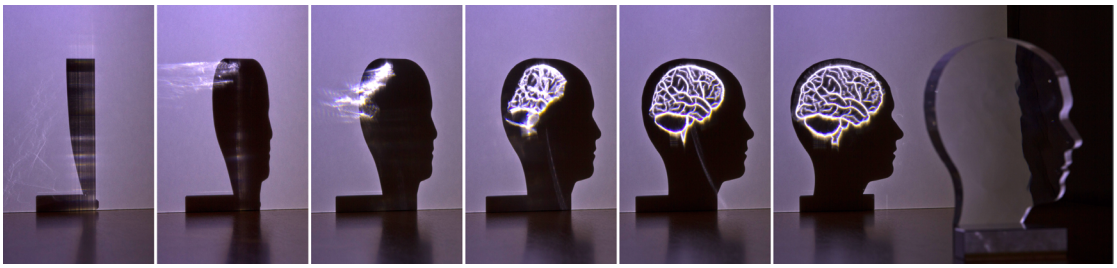


Figure 5.1: Caustic Brain: Our algorithm computes a 3D surface that refracts uniform light to focus on sharp intensity lines that sketch a human brain. The physical prototype shown on the right has been fabricated in transparent acrylic with a CNC milling machine. The photographs illustrate how the caustic image evolves as the acrylic piece is rotated into position (see also Figure 5.12 and accompanying video).

The previous two chapters dealt with the problem of finding methods of fabrication for 3D models. The first included artistic input as the structure of the outputted model, as the internal structure has a great influence of the aesthetics of the piece. Both methods produced pieces that would be difficult, or at least tedious, to build without computation. The constraints in each case also came directly from the fabrication and the building process. However, performative constraints can also be applied that do not come directly from the fabrication but secondary effects of it. The interplay of light and material is one such "side effect." But what if using computation that side effect can be used as a primary design driver?

In this chapter, we present a new algorithm for computational caustic design. Our algorithm solves for the shape of a transparent object such that the refracted light paints a

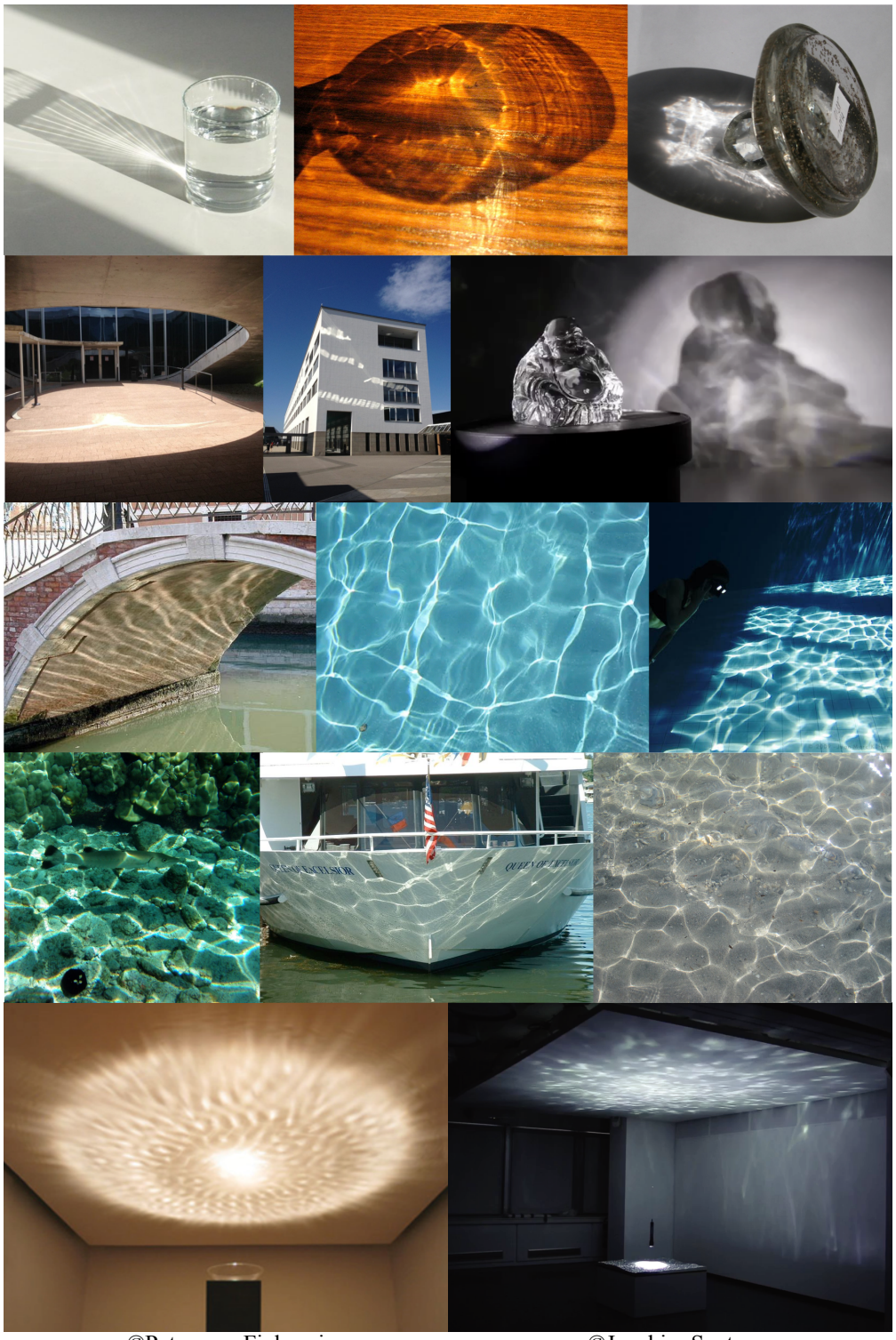
desired caustic image on a receiver screen. We introduce an optimal transport formulation to establish a correspondence between the input geometry and the unknown target shape. A subsequent 3D optimization based on an adaptive discretization scheme then finds the target surface from the correspondence map. Our approach supports *piecewise smooth* surfaces and non-bijective mappings, which eliminates a number of shortcomings of previous methods. This leads to a significantly richer space of caustic images, including smooth transitions, singularities of infinite light density, and completely black areas. We demonstrate the effectiveness of our approach with several simulated and fabricated examples.

5.1 Foreword

The interplay of light and form is fundamental in how we perceive the world. Reflective or refractive objects exhibit particularly rich interactions with light, often creating fascinating caustic effects. These effects, which we call caustics, are even found in everyday life such as light shining through a glass or reflecting off a metal bowl. While mostly accidental and seemingly random in everyday objects, artists have explored these caustic patterns for enticing light installations (see Figure 5.2). However, this deliberate use of caustics typically follows a simple trial and error design approach. Manually controlling the appearance of caustics is notoriously difficult, as slight changes to the specular surface can have large, non-local effects on the created caustic image. Objects such as the one shown in Figure 5.1 are virtually impossible to create with traditional means of shaping or sculpting a refractive material.

Instead, we follow a recent line of research that proposes computational solutions to approach this challenging inverse light transport problem [FDL10, PJJ+11, KEN+12, YIC+14]. We address fundamental shortcomings of these previous methods and propose the first solution capable of computing the shape of refractive objects that cast controlled, highly detailed caustic images of high contrast.

We introduce a new optimization algorithm for inverse caustic design based on optimal transport. The optimal transport formulation in combination with an adaptive Voronoi discretization scheme enables a significantly broader range of target light distributions than previous methods. Our solution supports continuous, *piecewise smooth* surfaces to allow easy fabrication, while enabling high-contrast target images including completely black areas, point and curve singularities of infinite light density in the target image, and non-uniform input light distributions with free boundaries.



©Petursson Finborgi ©Joachim Sauter

Figure 5.2: Caustics created by everyday objects and art installations (bottom row).

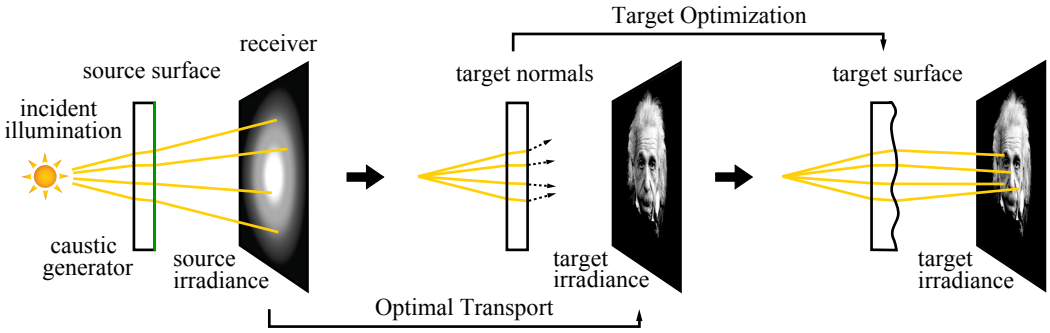


Figure 5.3: Processing pipeline. We first compute the initial source irradiance distribution on the receiver screen from the incident illumination. The optimal transport algorithm then finds a mapping to the target distribution from which we obtain the target normals on the source surface. The target optimization solves for the surface that best matches these normals. (Photo by Philippe Halsman © Philippe Halsman Archive)

These features, that have not been shown in any previous method, significantly expand the creative possibilities of computational caustic design and enable caustic lighting for a variety of application domains, such interior and product design, lighting, architecture, or art. We present several prototypes that demonstrate that physical realizations of our caustic generators well match the predicted behavior. Controlling caustics offers a number of exciting new design possibilities in a variety of fields, such interior and product design, architecture, or art.

5.2 Overview

We first introduce some terminology and give a high-level overview of our approach as illustrated in Figure 5.3. We assume we are given the initial geometry refractive object, the *caustic generator*, and the incident illumination on that object. The user also specifies the position and orientation of a *receiver* screen on which the desired caustic image should appear. The receiver is typically a diffuse planar surface of arbitrary orientation. The incident illumination is refracted through the initial geometry to create an irradiance distribution on the receiver that we call the *source irradiance* E_S. Our goal is to determine the shape of the caustic generator such that the resulting irradiance distribution on the receiver matches a desired *target irradiance* E_T provided by the user.

To formulate our optimization we make several simplifying assumptions. We use a geometric model of optics with perfect specular scattering. To create the desired caustic

image, we only modify a single scattering event for each light path, i.e we change the shape of a single refractive surface that we call the *source surface*. The optimized surface computed by our algorithm is the *target surface*. Light rays should arrive at each point of the source surface from a single direction only, which requires idealized illumination such as parallel light or light emitted by point sources. We comment on the effect of area light sources in Section 5.6.

Our algorithm first calculates the source irradiance E_S by raytracing the incident illumination through the source geometry. Then we compute an optimal transport map (OTM) from source to target on the receiver screen. This map encodes how the distribution of light needs to be modified to obtain the desired target image. More specifically, we compute a target position on the receiver for each light ray that leaves the source surface. Using Snell's law we can then determine appropriate normals on the source surface that can be integrated to obtain the optimized target surface.

We first describe how to define the target distribution in Section 5.3. The subsequent sections discuss the core components of our algorithm: Section 5.4 explains how to compute an OTM using an optimization on a power diagram, while Section 5.5 describes an iterative optimization algorithm to compute the 3D surface of the caustic generator from the OTM. In Section 5.6 we present several simulated and physical results and provide an evaluation and comparison of our method with previous work. We conclude with remarks on future research direction in Section 5.7.

5.3 Specifying the Target

As described in detail below, we use an optimal transport formulation to map the source irradiance E_S to the target E_T in order to compute the target positions on the receiver for each refracted light ray. In this setup, source and target are more conveniently represented as *radiant flux measures* Φ_S and Φ_T, respectively. A measure is a function that assigns a non-negative real number to subsets of a domain, satisfying certain properties such as non-negativity and countable additivity [Bog06]. We make use of this representation in Section 5.4.

Flux Φ and irradiance E are related as $\Phi(\Omega) = \int_\Omega E(x,y)\,dx\,dy$ for any subset $\Omega \subseteq \mathbb{R}^2$. Let U be the union of the supports of Φ_S and Φ_T on the receiver, i.e. the smallest closed set such that $\Phi_S(\mathbb{R}^2 \setminus U) = \Phi_T(\mathbb{R}^2 \setminus U) = 0$. In the following, we drop the subscript for a radiant flux that can refer to either Φ_S and Φ_T.

Singularities. One of the most fascinating features of caustics is the intense concentration of brightness when light is sharply focused onto points or curves. To capture this effect we support point and curve singularities of infinite light density in our target distribution Φ_T. These can be represented using point and line Dirac delta distributions [ZZ12]. We define three types of base functions Φ_0, Φ_1, and Φ_2 for point singularities, curve singularities, and area distributions, respectively. We specify point and curve singularities through an SVG file and area distributions as either an SVG file or a PNG image. These base functions can then be combined arbitrarily to build the desired target distribution Φ_T.

A point singularity δ is specified by a position $\mathbf{x}_\delta \in U$ and a flux Φ_δ, such that

$$\Phi_0(\Omega) = \begin{cases} \Phi_\delta & \text{if } \mathbf{x}_\delta \in \Omega \\ 0 & \text{otherwise.} \end{cases}$$

A curve singularity γ is represented by a curve $c : [0, L] \to \mathbb{R}^2$ and a curve density function $f : [0, L] \to \mathbb{R}$, where c is parameterized with respect to arc length and L is the total length of c. Then Φ_1 is defined as

$$\Phi_1(\Omega) = \int_0^L f_\Omega(t)\,dt,$$

where is f_Ω the restriction of f onto Ω:

$$f_\Omega = \begin{cases} f(t) & \text{if } c(t) \in \Omega \\ 0 & \text{otherwise.} \end{cases}$$

Finally, we define

$$\Phi_2(\Omega) = \int_\Omega E(x, y)\,dx\,dy,$$

where $E : U \to \mathbb{R}$ is an integrable 2D irradiance function.

Target. Let $\{\delta_1, \ldots, \delta_{N_\delta}\}$ be a set of N_δ point singularities and $\{\gamma_1, \ldots, \gamma_{N_\gamma}\}$ a set of N_γ curve singularities. The target flux function Φ_T is then represented by the combination of 0, 1, and 2-dimensional integral functions as

$$\Phi_T(\Omega) = \sum_{i=1}^{N_\delta} \Phi_0^i(\Omega) + \sum_{j=1}^{N_\gamma} \Phi_1^j(\Omega) + \Phi_2(\Omega). \tag{5.1}$$

We model curve singularities with piecewise linear representations. Unlike previous work, these singularities allow infinite light density. To represent the 2D irradiance function E we support a pixel grid of intensity values or a vector representation. Note that both the source and the target distributions can contain regions of zero intensity anywhere in U. In particular for the target, more than one singularity is not possible in previous methods that use continuous generator surfaces. Continuous surfaces inherently interpolate the rays and generate streaks and therefore discontinuous surfaces with sharp creases are necessary (see Figure 5.4).

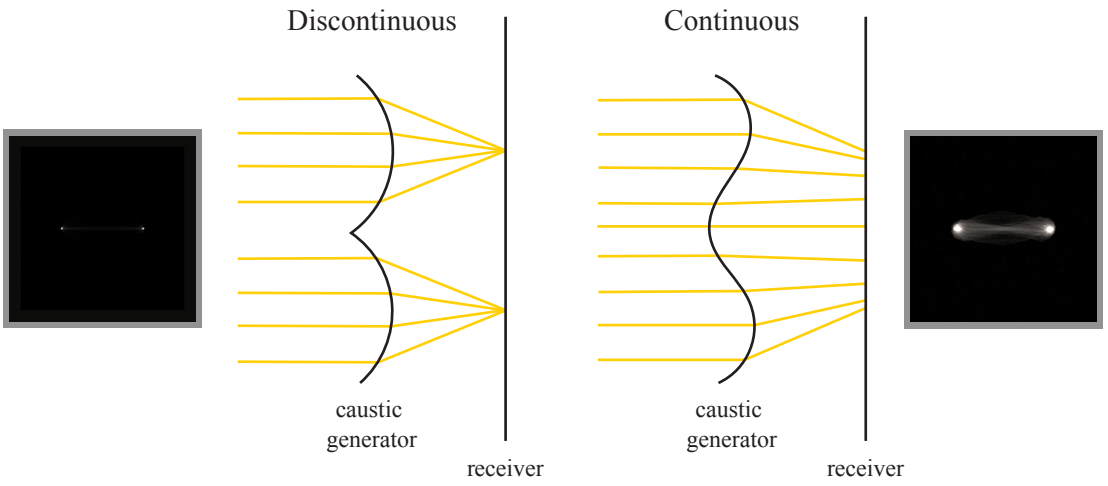


Figure 5.4: For a target of two points, a discontinuous surface with is able to distribute light evenly to the two points (left; in profile). A continuous surface will produce streaks and interpolate between the two points.

5.4 Optimal Transport

The goal of computational caustic design is to redirect light rays such that the desired target light image is drawn on the receiver screen. Since we assume a mostly diffuse receiver that scatters light equally in all directions, the perception of the caustic image depends mainly on the flux density or irradiance at each point, but is largely independent of the direction of the incoming rays hitting the screen. This means that we can formulate the optimization on the target positions of each light ray on the receiver. In other words, we need to answer the question: How do we have to move the initial source positions of each ray on the receiver towards new target positions, such that the overall irradiance distribution matches the target E_T as closely as possible? We solve this problem using an optimal transport formulation to compute the target positions. As we discuss below, optimal transport is ideally suited to handle discontinuities and

singularities, while maximizing smoothness to obtain high caustic image quality.

5.4.1 Continuous Optimal Transport

We briefly review the basic concepts of optimal transport that are most relevant for our approach. For an extensive review we refer to [Vil09]. Optimal transport is concerned with finding a mapping between two probability measures, a source measure μ_S and a target measure μ_T. A 2D *transport map* from μ_S to μ_T is a function $\pi : U \to U$ for some domain $U \subseteq \mathbb{R}^2$, such that $\mu_S(\Omega) = \mu_T(\pi(\Omega))$ for every subset $\Omega \subseteq U$. We can define a global transport cost for π as

$$c(\pi) = \int_U \|x - \pi(x)\|^2 d\mu_S(x).$$

For this ℓ_2 cost measure it has been shown that there exists a unique *optimal transport map* (OTM) that is a global minimizer of the total transport cost [Vil09]. Minimizing the transport cost in our context means that overall the directions of the refracted light rays are modified as little as possible. This minimizes the change in curvature of the target surface and ensures that no foldovers are introduced in the mapping. Among other benefits (see discussion below), this simplifies physical fabrication and helps avoid optical deficiencies such as internal reflections.

To apply the optimal transport formalism to our problem, we need to transform our radiant flux measures Φ_S and Φ_T into probability measures, i.e. measures of total mass of one. This can be easily achieved as

$$\mu_S(\Omega) := \frac{\Phi_S(\Omega)}{\Phi_S(U)}, \qquad \mu_T(\Omega) := \frac{\Phi_T(\Omega)}{\Phi_T(U)}, \tag{5.2}$$

where $\Phi_S(U) = \Phi_T(U)$ is the total mass of Φ, assuming that no light is lost due to absorption.

5.4.2 Discrete Optimal Transport

Our computation of the discrete OTM follows the algorithm introduced by Aurenhammer et al. [AHA98] and later extended and improved by Merigot [Mér11]. A similar approach has been proposed by de Goes et al. [dGBOD12].

We assume that the incident illumination is represented as a triangle mesh on the source surface, where each vertex carries an incoming direction and an intensity value. This

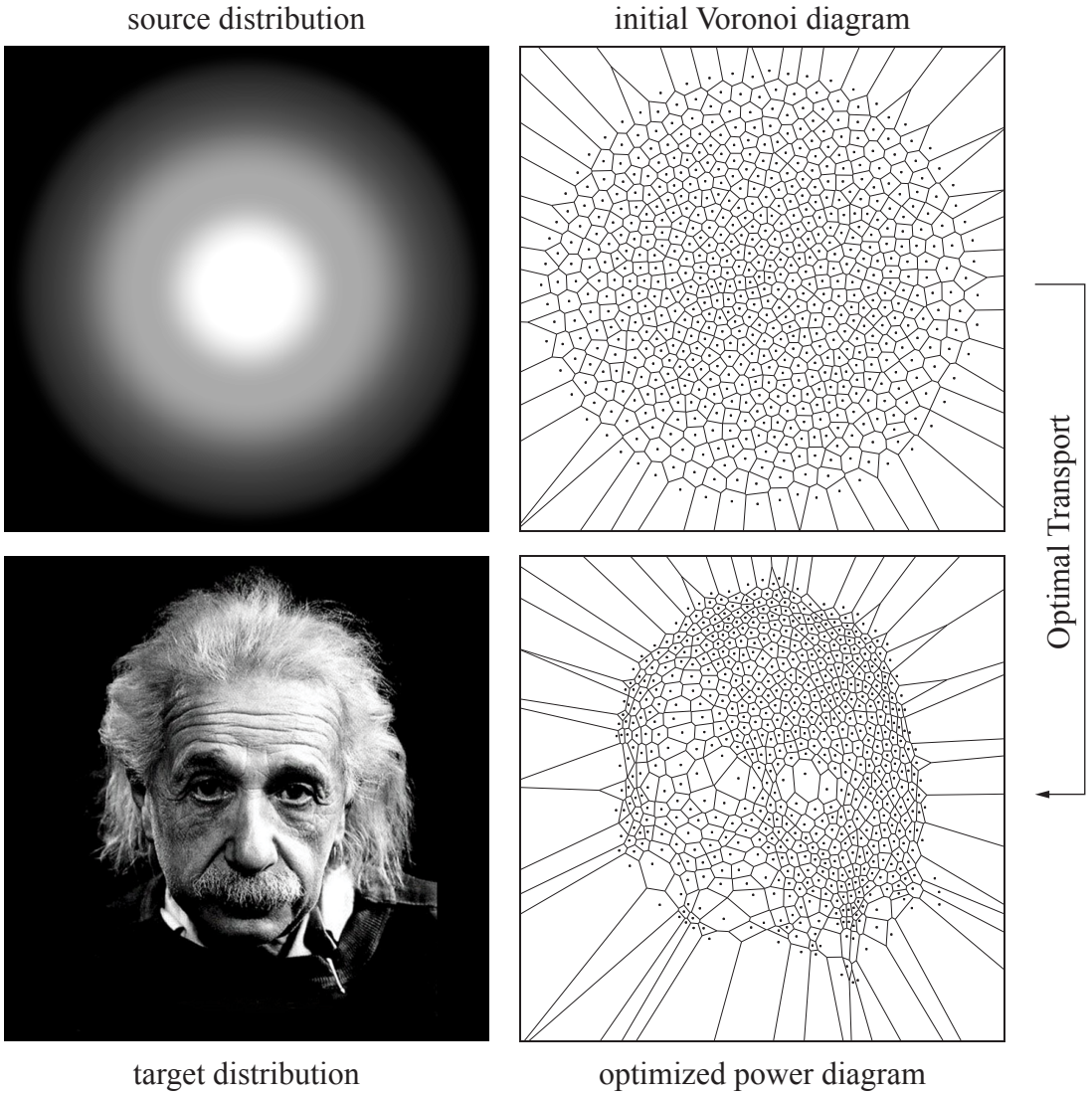


Figure 5.5: The source irradiance distribution is sampled using Lloyd sampling to obtain the initial Voronoi diagram, where each Voronoi cell approximately receives the same irradiance (low resolution for illustration). The optimization then computes the weights of the corresponding power diagram that best matches the target distribution. The dots show the centers of mass of the source and target distribution, respectively, integrated over the cells. (Photo by Philippe Halsman © Philippe Halsman Archive)

triangle mesh is ray-traced through the source surface onto the receiver screen to obtain a piecewise linear representation of the source irradiance E_S. Here we assume that the refraction through the source surface does not create any fold-overs or singularities.

We resample E_S such that each sample roughly represents the same amount of flux

to best exploit the degrees of freedom in the OTM optimization. For this purpose, we apply Lloyd sampling on E_S to obtain a set $\mathscr{S} = \{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$ of *sites* $\mathbf{s}_i \in U$. The domain U is then discretized using a Voronoi diagram of $\mathscr{S}$. The goal is to determine how each Voronoi cells need to be deformed and moved, such that its flux is distributed to match the desired target distribution. This can be achieved by representing the target distribution as a power diagram (weighted Voronoi diagram) on the sites S and finding suitable weights (see Figure 5.5).

Let P_ω be the power diagram of $\mathscr{S}$ with a set of weights $\omega = \{\omega_1, \ldots, \omega_n\}$. When all weights ω_i are zero, the power diagram coincides with the Voronoi diagram, which we thus denote as P_0. Let C_i^ω be the power cell of P_ω associated with site $\mathbf{s}_i$. Aurenhammer and colleagues [AHA98] have shown that there is a unique assignment of weights such that $\mu_S(C_i^0) = \mu_T(C_i^\omega)$ for all i.

Recalling the definitions of Equations 5.1 and 5.2, this means that for a suitable set of weights ω, light refracted by the initial base geometry onto the Voronoi cell C_i^0 on the receiver will be redirected to the power cell C_i^ω of P_ω. The ratio of areas of C_i^0 and C_i^ω accounts for the relative difference of intensity in the source and target.

Merigot [Mér11] showed that the unknown weight vector ω can be found as a global minimizer of the convex function

$$f(\omega) = \sum_{\mathbf{s}_i \in S} \left(\omega_i \mu_S(C_i^0) - \int_{C_i^\omega} (\|\mathbf{x} - \mathbf{s}_i\|^2 - \omega_i) d\mu_T(\mathbf{x}) \right). \tag{5.3}$$

This result is fundamental for our algorithm as it ensures that the OTM can be found by a suitable gradient descent scheme. It turns out that the corresponding gradients are simply the differences of the integrated source and target densities, i.e.

$$\frac{\partial f}{\partial \omega_i} = \mu_S(C_i^0) - \mu_T(C_i^\omega). \tag{5.4}$$

We use an L-BFGS solver [LN89] to minimize the objective function f of Equation 5.3, following the multi-scale strategy proposed by Mérigot [Mér11]. We refer to this paper for detailed derivations and implementation details. In all our examples the number of points is scaled by a factor of four between two different levels of the multi-scale solver. We use the CGAL library [Yvi13] with exact arithmetic to compute the power diagrams. Recently, Bruno Lévy [Lé14] has been able to use the multi-scale method to solve optimal transport in 3D. Similar improvements can potentially be used to speed up our formulation.

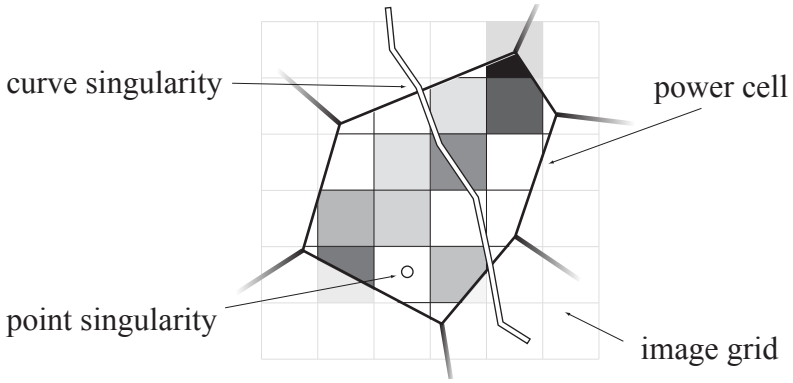


Figure 5.6: Integration of the target measure on a power cell. Curve singularities are approximated by piecewise linear segments.

Singularities. Merigot's solution maps a source density to a finite set of point sites. We use an inverse formulation, i.e. we map a set of sites, the samples on the source distribution, to the density defined by the target measure. This allows us to naturally integrate singularities in the target measure by adapting the calculations of the integrals in Equations 5.3 and 5.4 (see Figure 5.6). We split the integrals over the target measure on each power cell into separate terms according to Equation 5.1 and evaluate each term separately. Point singularities are trivial to evaluate. For curve singularities, we approximate each curve by a polygon segment and the corresponding density by a piecewise linear function defined over the polygon, leading to a simple analytical formula for the integration (see Section 5.3). To evaluate integrals over pixel areas and similarly for filled polygons, we first compute the intersection of each pixel with the cell to adapt the pixel boundaries appropriately. We assume constant density for each pixel area and apply Green's theorem to transform the area integral into a simpler boundary integral. A benefit of our optimal transport formulation compared to previous methods [KEN+12, YIC+14] is that the OTM is not required to be bijective. This means, for example, that we can focus light going through a certain area of the generator onto a single point or curve, thus creating singularities of infinite light density.

Discontinuities. The regularity of optimal transport has been studied extensively, see [DPF13] for a recent survey. Essentially, OTMs are continuous for a quadratic cost function like the one we use, if the probability measures are sufficiently regular and the target convex. Nevertheless, discontinuities arise naturally in optimal transport for non-convex or high-contrast targets, see [CJL+13] for a sufficient condition for discontinuity in planar OTMs. Adjacent regions in the source can be mapped to distant regions on the target, which leads to discontinuities in the normals of the target surface. These discontinuities are in fact necessary when aiming at completely black interior regions,

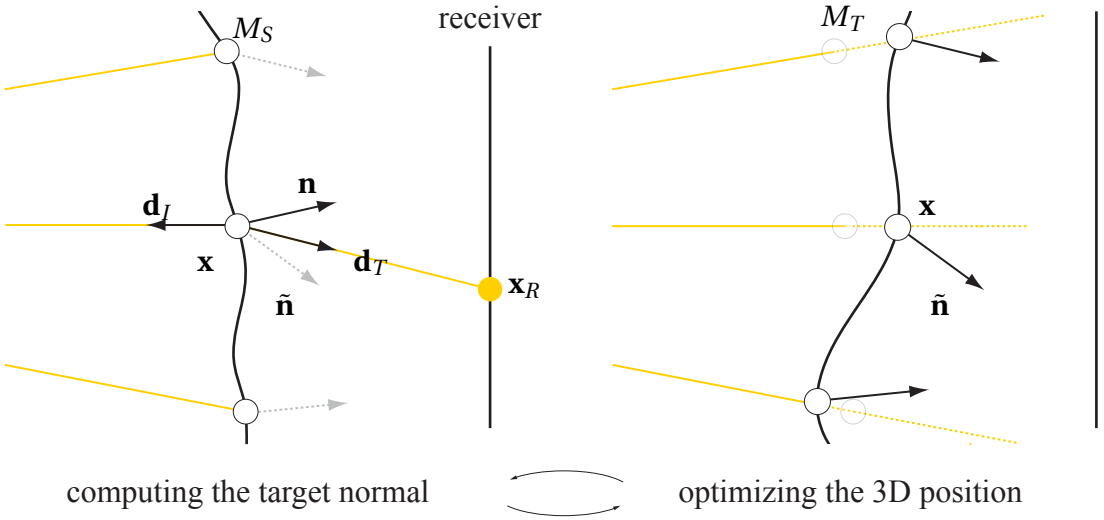


Figure 5.7: Two-stage target optimization. We first compute the target normal $\tilde{\mathbf{n}}$ for each vertex (left), then solve for the vertex position $\mathbf{x}$ to obtain the target surface that matches these normals (right). Because $\mathbf{x}$ changes, the target direction $\mathbf{d}_T$ and consequently the normal $\tilde{\mathbf{n}}$ need to be updated, hence both stages are iterated.

for example (see Figure 5.10). This additional flexibility to introduce discontinuities in the mapping is a major benefit of optimal transport, as it strongly reduces distortion artifacts commonly observed in globally smooth methods such as [KEN+12, YIC+14] (see also Figures 5.13 and 5.14).

5.5 Target Optimization

The OTM provides us with a discrete mapping between source and target irradiance distributions. More precisely, it provides the positions where rays leaving the source surface *should* intersect the receiver. The goal of the target optimization is now to find the corresponding target surface that refracts the rays towards these target positions. For this purpose we discretize the target surface with a triangle mesh that is initialized on the source surface. We compute target normals for each vertex, using Snell's law, from the target ray directions derived from the OTM. The optimization then moves the vertices to best match these normals while respecting the desired flux densities. Since modifying the vertex positions changes the target directions, we iterate this process (see Figure 5.7).

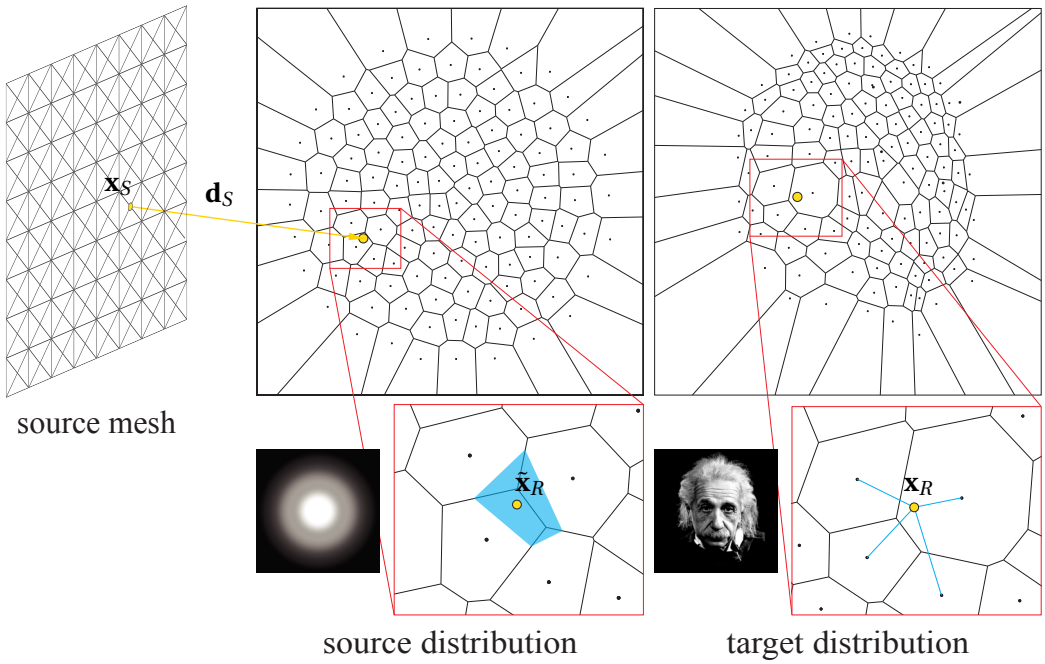


Figure 5.8: Natural neighbor interpolation of the OTM onto the vertices of the source mesh. First we obtain $\tilde{\mathbf{x}}_R$ as the intersection with the receiver plane of the ray leaving the source surface at $\mathbf{x}_S$ in direction $\mathbf{d_S}$. Inserting this point into the Voronoi diagram of the source distribution yields the blue cell. The fraction of area of each original Voronoi cell that the blue cell covers defines the interpolation weights for computing the target position $\mathbf{x}_R$ from the corresponding centroids of the target power diagram (right).

OTM Interpolation. As illustrated in Figure 5.5, the power diagram adapts to the target distribution and can thus be highly non-uniform. However, for light transport simulation or fabrication, a uniform sampling of the target surface is preferable. We therefore represent the target surface by a uniform triangle mesh M_T of user-specified resolution. Let $v = \{\mathbf{x}, \mathbf{n}, \mathbf{d}_I, \mathbf{x}_R\}$ be a vertex of M_T, where $\mathbf{x}$ is the vertex position, $\mathbf{n}$ the corresponding surface normal, $\mathbf{d}_I$ the direction of the incoming ray, and $\mathbf{x}_R$ the intersection of the outgoing ray with the receiver; see Figure 5.7. The positions $\mathbf{x}$ and incoming ray directions $\mathbf{d}_I$ are initialized from the source surface and incoming illumination, respectively. We retrieve the target position $\mathbf{x}_R$ from the OTM. Recall that the OTM defines a point-wise correspondence between the weighted centroid of each Voronoi cell and the corresponding power cell. To obtain the target position $\mathbf{x}_R$ we interpolate the computed receiver positions onto M_S using natural neighbor interpolation [Sib81], as shown in Figure 5.8. Since natural neighbor interpolation is only defined within the convex hull of the centroids, we extrapolate by moving the centroids of boundary cells onto the boundary. This causes a slight deformation at the boundary, but the effect

diminishes as the OTM resolution increases.

Given an incoming light vector $\mathbf{d}_I$ and a target direction $\mathbf{d}_T = (\mathbf{x_R} - \mathbf{x})/\|(\mathbf{x_R} - \mathbf{x})\|$, we compute the desired target surface normal using Snell's law as $\tilde{\mathbf{n}} = \mathbf{d}_I + \eta\mathbf{x}/\|\mathbf{d}_I + \eta\mathbf{d}_T\|$, where η is the ratio of the refraction indices of the two media. We refer to [KP12] for a derivation of this formula.

3D Optimization. Even though the OTM is curl-free, the normals derived from the corresponding targets are not necessarily integrable, due to the non-linearity introduced by Snell's law. To compute the target surface, we thus formulate an iterative optimization that solves for the target vertex positions by minimizing the following compound energy:

$$\arg\min_{\mathbf{x}} \mathbf{w} \cdot [E_{\text{int}}, E_{\text{dir}}, E_{\text{flux}}, E_{\text{reg}}, E_{\text{bar}}], \tag{5.5}$$

where $\mathbf{w}$ is a weighting vector. The integration energy aligns the vertex normals $\mathbf{n}$ of M_T with the target normals $\tilde{\mathbf{n}}$ derived from the OTM:

$$E_{\text{int}} = \sum_{v \in M_T} \|\mathbf{n} - \tilde{\mathbf{n}}\|_2^2, \tag{5.6}$$

where $\mathbf{n}$ is computed by averaging the normals of incident triangles weighted by the incident angles [BKP+10, Pg.41]. Previous techniques constrain the vertices of M_T to lie on the associated incoming ray direction and therefore only need to solve for a scalar vertex displacement. As illustrated in Figure 5.9, this can be problematic, because the mesh cannot adapt to sharp creases caused by discontinuities in the normal field. We therefore perform the optimization over all spatial coordinates of the mesh to allow vertices to slide along the surface to better represent crease lines. However, to maintain consistency with the OTM, we need to ensure that vertices do not deviate too much from the incoming ray direction. This can be achieved with a penalty term

$$E_{\text{dir}} = \sum_{v \in M_T} \|\mathbf{x} - \mathbf{proj}_{(\mathbf{x}_S, \mathbf{d}_I)}(\mathbf{x})\|_2^2, \tag{5.7}$$

where $\mathbf{x}_S$ is the position of vertex v on the source surface. The projection operator **proj** returns the point on the line $(\mathbf{x}_S, \mathbf{d}_I)$ closest to $\mathbf{x}$. In addition, we need to ensure that the flux over triangle t of M_T remains constant, because the OTM was computed according to this flux. Therefore, we insert a flux preservation energy for each triangle:

$$E_{\text{flux}} = \sum_{t \in M_T} \|\Phi_T(t) - \Phi_S(t_S)\|^2, \tag{5.8}$$

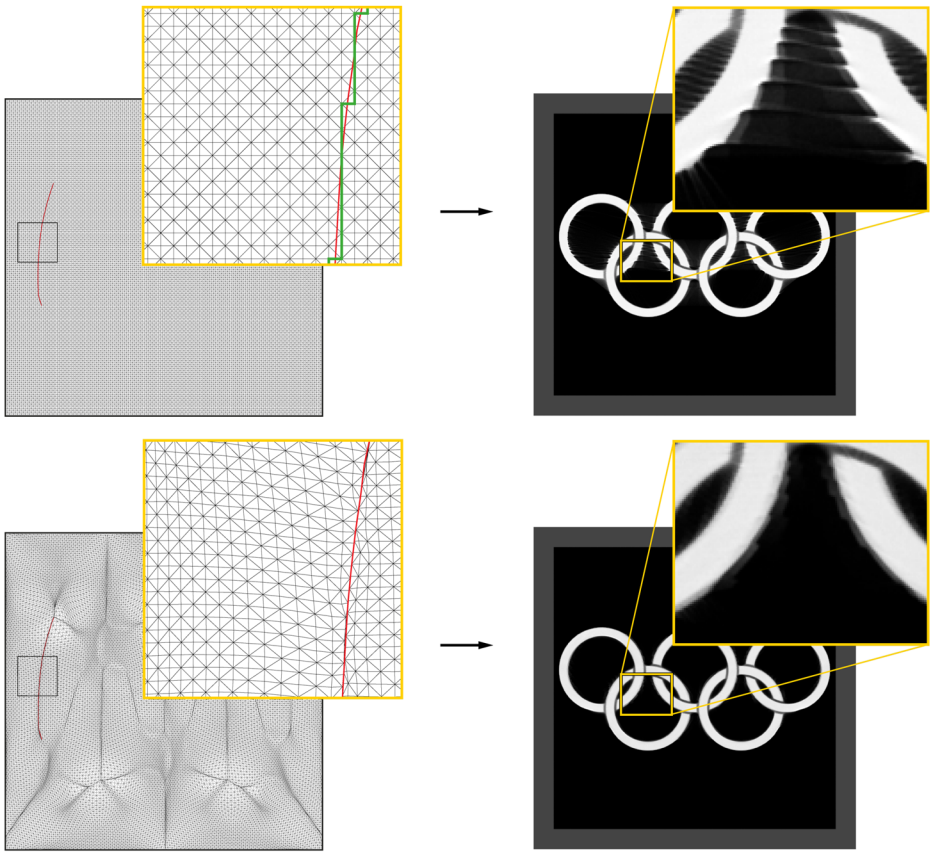


Figure 5.9: Benefit of 3D integration. The result of our optimal transport algorithm is integrated on a regular height field, as used in previous integration methods [KEN+12, YIC+14] (top row). Since the grid cannot align with the sharp creases produced by discontinuities in the OTM, artifacts appear at high contrast transitions. Our full 3D integration scheme largely avoids these artifacts by properly aligning mesh edges to the creases (bottom row). (SVG source: Wikipedia. The Olympic Rings are ® International Olympic Committee)

where t_S is the triangle corresponding to t on the source surface. To maintain well-shaped triangles we add the regularization term:

$$E_{\text{reg}} = \|\mathbf{LX}\|_2^2, \tag{5.9}$$

where the vertices of M_T are stacked in $\mathbf{X}$, and $\mathbf{L}$ is the corresponding combinatorial Laplacian matrix [BKP+10]. Finally, we introduce an additional barrier energy to ensure the physical realizability of the piece by preventing the surface to fall beyond a

certain distance d_{TH} from the receiver:

$$E_{\text{bar}} = \sum_{v \in M_T} \| f_{bar}(\mathbf{n}_R \cdot (\mathbf{x} - \mathbf{x}_R)) \|^2 \tag{5.10}$$

$$f_{bar}(x) = \max(0, -\log((1-x) + d_{\text{TH}})) \tag{5.11}$$

where $\mathbf{n}_R$ is the normal of the receiver plane.

The pseudocode of the target optimization algorithm is provided below.

```
// INPUT:
//   X := vertex positions of M_S
//   R := optimal transport map
// OUTPUT:
//   X := vertex positions of M_T
function X = TARGET-OPTIMIZATION(X, R)
    X_R = OTM-Interpolation(X,R)
    while NotConverged() do
        D_T = normalize(X_R − X)
        Ñ = Fresnel-Mapping(X, D_T)
        X = Normal-Integration(X, Ñ)
    end while
end function
```

We solve Equation 5.5 using the auto-differentiation Levenberg-Marquadt optimization offered by the *Ceres* framework [AMO13]. The optimization converges after a few outer iterations of Fresnel Mapping followed by 3D optimization.

For all our examples, we set the weights for E_{int} and E_{bar} to 1. For the EINSTEIN, OLYMPIC, and BRAIN models we use 1×10^{-6} for the weight of E_{dir} and no flux energy, while the SIGGRAPH and LENA models are computed with 1×10^{-4} for E_{dir} and 1×10^{-3} for E_{flux}. The results are not particularly sensitive to the choice of these parameters. More difficult to tune is the weight for regularization term E_{ref} (we select values between 1×10^{-6} and 1×10^{-4} in our examples). If chosen too low, triangle inversions might occur that lead to an inconsistent surface. On the other hand, if the weight is too high, we deviate from the optimal solution, which introduces distortions in the caustic image. In the future we want to investigate more sophisticated regularization terms that prevent triangle inversions without negatively affecting the other objectives.

Alternative using alternating minimization As an alternative to Ceres, we can use an alternating minimization formulation similar to that of Shape-Up [BDS+12,

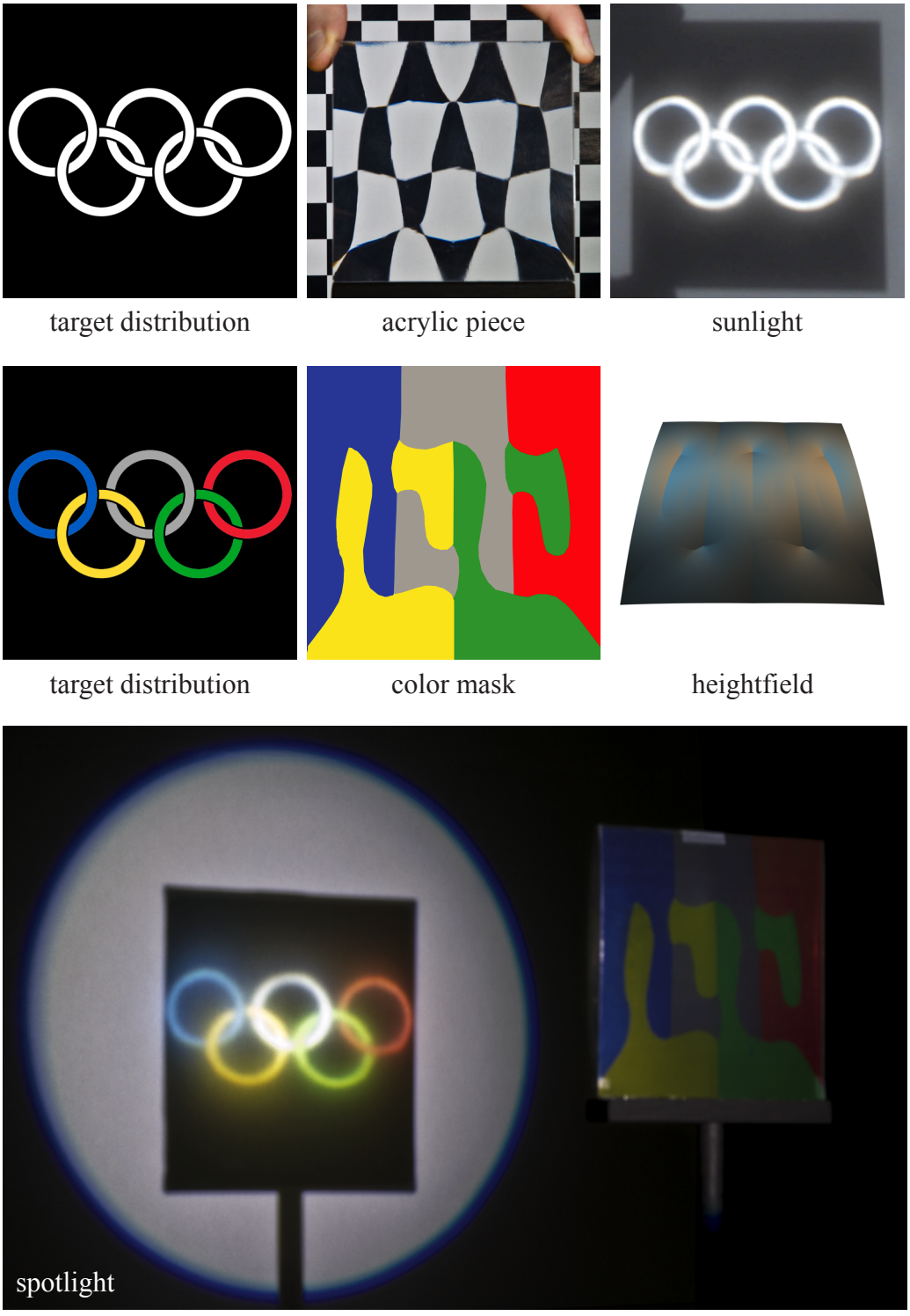


Figure 5.10: A caustic image of the Olympic rings photographed under sunlight (top right) and a spotlight (bottom) with a color mask computed from the OTM.

BML+14]. This approach is similar to the method of Xie et al. [XZWC14]. In this version, we convert the target vertex normals to face normals which we look up from the OTM at the centroid of each face. We reformulate each of the energies from the section below as a constraint and minimize a sum of energies, where each term has the form:

$$W(\mathbf{q},\mathbf{x}) = w||\mathbf{A}\mathbf{q} - \mathbf{B}\mathbf{x}||_2^2 + \delta_{\mathbf{C}}(\mathbf{x}), \tag{5.12}$$

where $\mathbf{A},\mathbf{B}$ are matrices, $\delta_{\mathbf{C}}$ is an indicator function which returns 0 if the constraint $\mathbf{C}$ is satisfied and ∞ if not, and w is a weight associated to each constraint. We can solve this in an alternating fashion by first fixing the $\mathbf{x}$ and solving for $\mathbf{q}$, and then fixing $\mathbf{q}$ and minimizing for $\mathbf{x}$. We reformulate each of the constraints in the previous section as follows.

For the integration energy, Equation 5.6, we first define the centroid $\mathbf{p}_i$ for each face $f_i \in M_T$ and find its normal $\mathbf{n}_i$ interpolating the OTM. For each face, we find the projections onto the constraint set $\mathbf{q}_{ij}$ where $0 \le j < |f_i|$. $|f_i$ denotes the number of vertices in f_i. To obtain $\mathbf{q}_{ij}$ we project the point of each vertex on the face onto the plane defined by the centroid and the normal, in the direction of the incoming light direction $\mathbf{d}_i$.

$$\mathbf{p}_i = \frac{1}{|f_i|} \sum_{v_j \in f_i \in M_T} \mathbf{x}_j \tag{5.13}$$

$$\mathbf{q}_{ij} = \langle \mathbf{p}_i - \mathbf{x}_j, \mathbf{n}_i \rangle \,/\, \langle \mathbf{d}_i, \mathbf{n}_i \rangle \, \mathbf{n}_i + \mathbf{x}_j \tag{5.14}$$

$$\mathbf{N} = \left(\mathbf{I}_{4\times 4} - \frac{1}{4}\mathbf{1} \right) \otimes \mathbf{I}_{3\times 3} \tag{5.15}$$

$$W_{\text{int}}(\mathbf{x}) = w_{\text{int}} \sum_{f_i \in M_T} ||\mathbf{N}\mathbf{q}(f_i) - \mathbf{N}\mathbf{x}(f_i)||_2^2 \tag{5.16}$$

Here $\langle \cdot,\cdot \rangle$ denotes an inner product, $\otimes$ is the Kronecker product, $\{\mathbf{q},\mathbf{x}\}(f_i)$ means the items of $\{\mathbf{q},\mathbf{x}\}$ corresponding to face f_i stacked.

We can use a similar formulation to Equation 5.7 since it is already formulated as a projection. We have a $\mathbf{q}_i$ for each $v_i \in M_T$.

$$\mathbf{q}_i = \mathbf{proj}_{(\mathbf{x}_{is},\mathbf{d}_i)}(\mathbf{x}_i) \tag{5.17}$$

$$W_{\text{dir}}(\mathbf{x}) = w_{\text{dir}} \sum_{v_i \in M_T} ||\mathbf{q}_i - \mathbf{x}_i||_2^2 \tag{5.18}$$

In case of freeform boundaries (e.g. Figure 5.12), we can introduce a term to project points on the boundary to the boundary curve, similar to the Equation 5.18, but projecting onto the closest line segment of the boundary curve orthogonally to the incoming light direction.

As for the area preservation term, we refer to [BML+14]. Finally, we can introduce a Laplacian term similarly to the previous section.

5.6 Results and Discussion

In this section we present several simulated results and physical prototypes computed by our optimization algorithm. We also evaluate the approximation quality with a ground truth example, compare our solution to previous methods, and comment on limitations of our approach. All light transport simulations have been generated using the physics-based rendering software *LuxRender*.

Figures 5.1 and 5.12 show a challenging example on a freeform domain. The high concentration of uniform light onto a complex network of singular curves of varying intensity and separated by black regions, is achieved through numerous discontinuities in the OTM. Figure 5.10 shows another high contrast caustic with completely black interior and exterior regions. The distortion of the checkerboard illustrates the strong refraction of the caustic generator. A simple extension allows for colored caustics. If the target distribution is wavelength-dependent, e.g. given by a color image, we can create a corresponding semi-transparent color filter on the caustic generator based on the OTM. The resulting color mask nicely illustrates how the incoming illumination is distributed towards the different rings. Note how no light is lost to the background in these examples (see also Figure 5.13). This kind of effect can only be achieved by adequate handling of discontinuities. Figure 5.11 combines a smooth image with high-intensity singularity lines. Although there are slight distortions due to fabrication errors, the result retains its high contrast and quality. The setup for each example is listed in Table 5.1.

Performance. Thanks to the multi-scale approach of the OTM optimization, we empirically observe roughly linear complexity in the number of input samples. However, each iteration requires re-computing the power diagram using exact arithmetic, which is computationally involved. For the example in Figure 5.10, it took 4 minutes for 16 thousand points, 25 minutes for 66 thousand points, and 95 minutes for 260 thousand

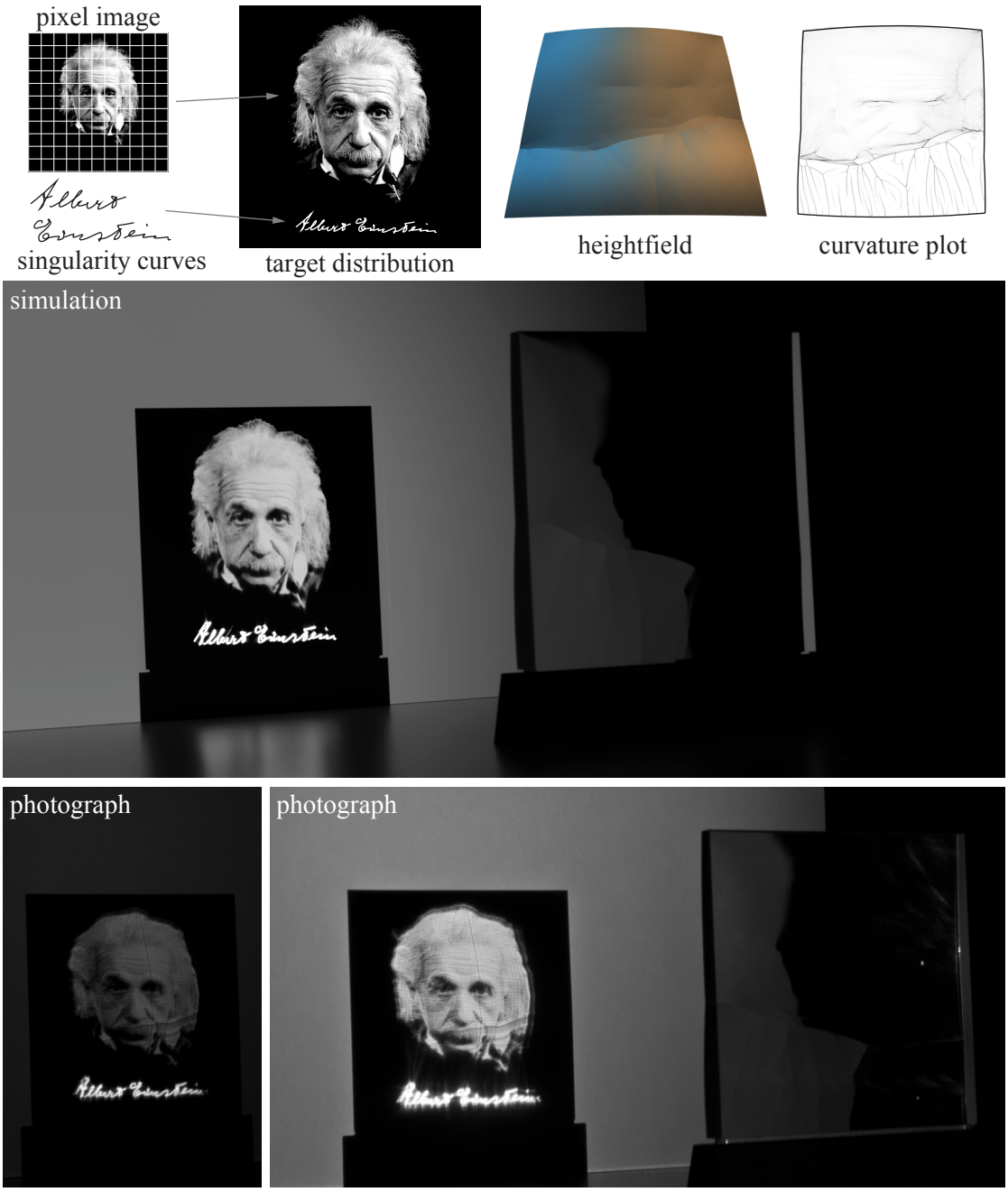


Figure 5.11: Signed portrait of Albert Einstein. A pixel image is combined with several singularity curves for the signature to define the target distribution. The total flux of the curves has been chosen as half the total flux of the image, which is reflected in the area distribution visible in the curvature plot. We use an exposure time of half a second (bottom left) and two seconds (bottom right) in the photographs to show the high dynamic range of the caustic image. (Photo by Philippe Halsman © Philippe Halsman Archive)

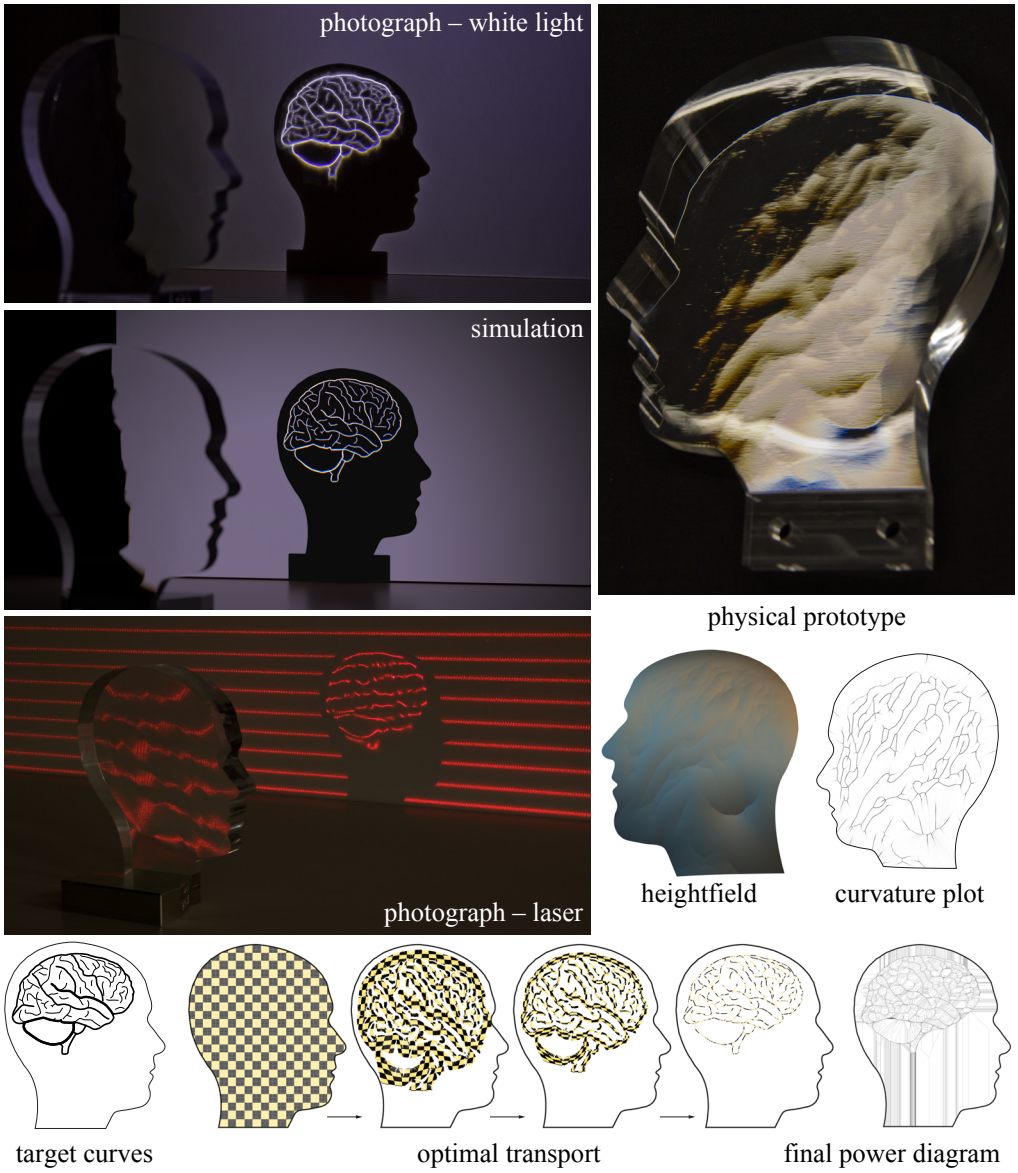


Figure 5.12: The caustic generator of the brain. The target distribution is composed of a number of singularity curves, where the line thickness indicates the relative flux density of each line. Computed from an initial uniform sampling, the final power diagram (1/4 the sample size for better readability) illustrates the highly non-uniform discretization necessary to match the target. The bottom-right row shows how a set of regularly sampled points is transported under the OTM. At left, a comparison of the physical prototype with a light transport simulation computed with LuxRender with roughly the same camera and light source parameters, and a photograph of horizontal laser beams passing through the piece demonstrating how light is refracted.

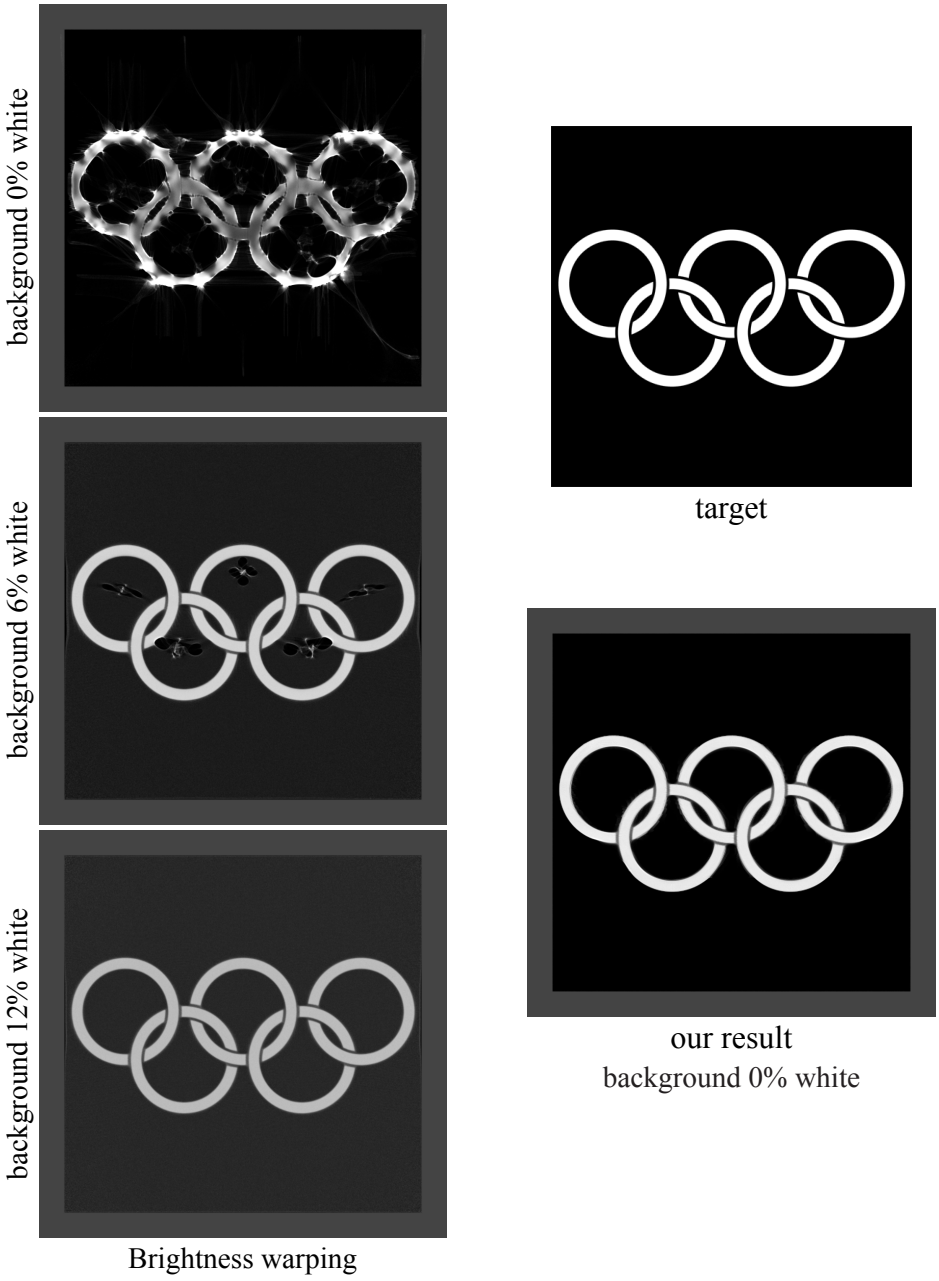


Figure 5.13: Comparison of our approach to the approach of Kiser et al. [KEN+12]. This method exhibits strong distortions for a black background (left). These artifacts can be reduced by brightening the background at the cost of reduced contrast. The artifacts disappear completely at about 12% white background, but now only 50% of the total incident illumination is focused on the rings. The gray border indicates the intensity of the uniform input illumination.

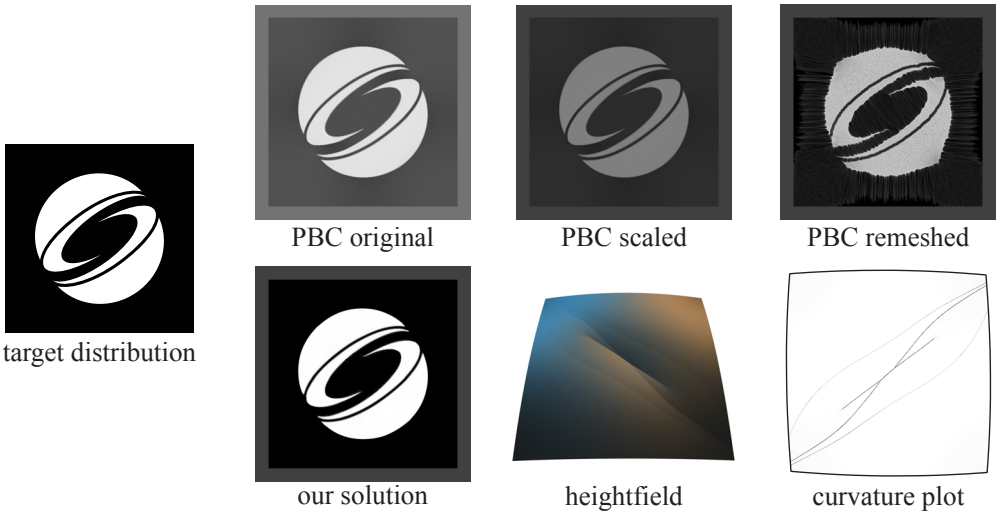


Figure 5.14: Comparison of our method to the Poisson-based continuous (PBC) method of Yue et al. [YIC+14]. Their algorithm needs to artificially reduce contrast to match the shape of the logo. The original image is taken from the paper. The scaled image matches the overall light intensity of our solution, as indicated by the gray border that shows the intensity of the uniform input illumination. An extension of PBC that dynamically remeshes the domain yields strong distortion artifacts. In contrast, our result achieves high caustic image quality. All PBC results have been produced by Yue et al.

points. The 3D target optimization takes between 3 minutes for a mesh of size 321×321 to about 15 minutes for a 641×641 mesh. All reported results are from a 2013 MacBook Pro with a 2.3GHz quad-core processor and 16GB of RAM.

The largest test case we ran took four hours of compute time to calculate the surface of a caustic generator of 1.5 million samples. More than 90% of the time is devoted to the optimal transport computation of which 99% of the time is spent recomputing the power diagram in each iteration. However, depending on the complexity of the target distribution, already significantly fewer samples suffice to achieve good results as listed in the table below.

Ground truth comparison. Figure 5.15 illustrates how our optimization approximates a ground truth result when increasing the resolution. For this example, uniform incident illumination on a circular domain is projected onto a uniform circular singularity curve centered at the origin. Through symmetry we see that under optimal transport each radial line is mapped to the corresponding closest point on the circle. For such a line, we can derive an analytic solution using Snell's law that can then be radially

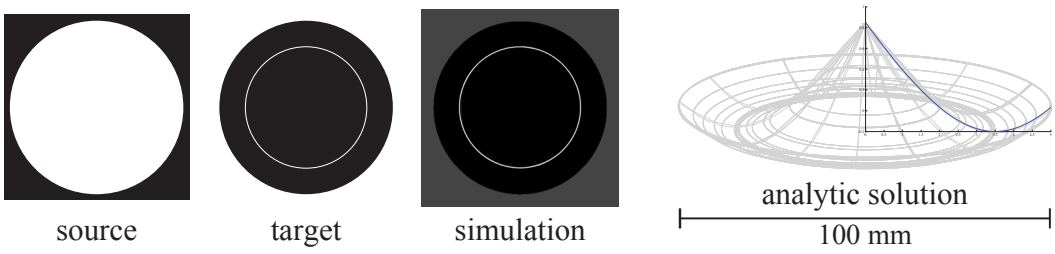
Image	EINSTEIN	OLYMPIC	BRAIN	SIGGRAPH	LENA
Mesh size	641^2	641^2	641^2	641^2	1281^2
OT samples	261,121	261,121	162,739	254,016	1,305,600
Physical size	10×10 cm	10×10 cm	11.5×13.5 cm	10×10 cm	10×10 cm
d_{TH}	30 cm	30 cm	40 cm	30 cm	10 cm
w_{int}	1	1	1	1	1
w_{dir}	1×10^{-6}	1×10^{-6}	1×10^{-6}	1×10^{-4}	1×10^{-4}
w_{flux}	0	0	0	0.001	0.001
w_{bar}	1	1	1	1	1
w_{reg}	2×10^{-4}	1×10^{-4}	2×10^{-4}	1×10^{-6}	1×10^{-6}

Table 5.1: The setup and optimization energy weights for each of the Caustic pieces.

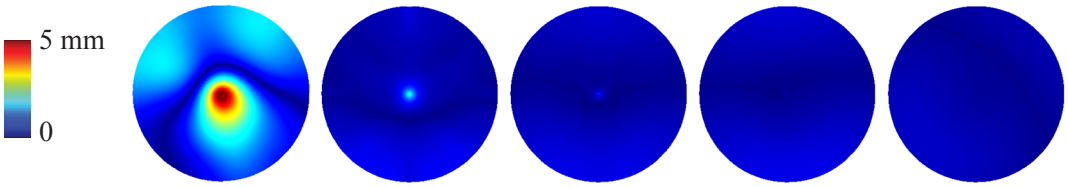
swept to obtain the ground truth 3D shape of the generator. As the results indicate, our solution quickly approaches to the ground truth under refinement.

Comparison to previous methods. Figures 5.13 and 5.14 show a comparison of our approach to the methods of [KEN+12] and [YIC+14], respectively. These methods achieve the highest quality caustic images to date. Both algorithms are similar in that they use fixed boundary conditions and enforce smoothness of the generator surface everywhere. While suitable for low-contrast images, where both achieve excellent results (see Figure 5.16), these constraints lead to visible artifacts when aiming for high-contrast images or black regions. These artifacts can only be avoided by artificially reducing contrast. In Figures 5.14 and 5.13 this is achieved by increasing the brightness of the background to a point where a substantial amount of light is "lost" for the actual caustic image. In contrast, our solution can harvest all light and supports completely black regions anywhere on the caustic. However, the adaptive discretization of the optimal transport algorithm comes at the price of increased computational overhead (see statistics above). The methods of [KEN+12] and [YIC+14] that optimize on a regular grid are about 10 times faster, and therefore may still be preferable for low contrast images.

Discussion and Limitations. A key aspect of our algorithm is that optimal transport *automatically* determines where discontinuities in the normal field should occur. Instead of having discontinuities everywhere as in [WPMR09, PJJ+11], or no discontinuities anywhere as in [KEN+12, YIC+14], our approach strikes a balance that achieves superior image quality, while not imposing unnecessary restrictions on the achievable caustic images.



Error vs. OTM resolution for source mesh of 80,957 vertices



#samples	45	208	1,042	11,996	82,369
max. error	6.1912	2.1206	0.8753	0.4636	0.3928
avg. error	1.1660	0.1902	0.1588	0.1666	0.1130

Error vs. source mesh resolution using analytic normals

vertices	5,152	20,348	80,957	322,899	1,289,432
max. error	1.5545	0.7513	0.3928	0.2064	0.1251
avg. error	0.3777	0.1909	0.1003	0.0795	0.0771

Linear blend of Voronoi diagram to power diagram for 208 samples

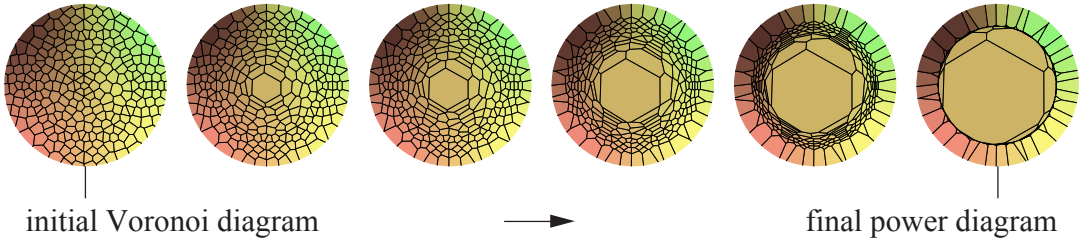


Figure 5.15: Ground truth evaluation. A disk of uniform light is focused onto a circular singularity curve by a hat shaped surface that can be computed in analytical form. All errors are in mm. The simulation shows the rendered caustic corresponding to an OTM resolution of 82,369 samples (top-right error plot).

Input image

[YIC+14]

[KEN+12]

Our method

Figure 5.16: For a smooth image like Lena, the performance of our algorithm is comparable to the state-of-the-art. For comparison, we use the input image as the basis for the gamma and global brightness of the other images. The top-right image has been provided by the authors of [Yue et al. 2014]. (Lena photo © Playboy Magazine)

While our optimization supports singularities in the target distribution, physical models will always deviate from the perfect surface and blur out singularities to finite areas (see Figures 5.1 and 5.12). Explicitly modeling singularities is still useful, because it provides a more principled and complete mathematical formulation and avoids having to manually select the spatial width of high-intensity curves in a pixel grid. This could lead

to excessive resolution of the input image and thus substantially increase computational cost.

The simplifying assumptions discussed in Section 5.2 incur a number of limitations of our approach. Area light sources, such as the sun, violate the assumption of a single incident light direction for each surface point. In general, this introduces blur in the caustic image, similar to the blur of shadow boundaries that cannot be avoided completely. Further blurring is introduced by imperfect specular scattering, machining imprecisions, and the necessary polishing process to remove milling path artifacts. Despite the accumulative nature of these effects, our prototypes convey that physical realizations maintain the overall quality of the target caustic images. An interesting direction for future work, especially once machining precision improves, is to consider the wave nature of light and study the possible resolution of caustics at the limit imposed by diffraction and incorporate partial coherence as in [LGX+13].

Our optimization does not take into account potential self-shadowing, internal reflections, or physical limits of refraction. If the angle of the incident illumination on the source surface becomes too shallow, or the target normal deviates too strongly from the source normal, artifacts will occur.

Our approach solves for one specific mapping defined by the unique optimal transport map. While this mapping has several important benefits in terms of regularity and smoothness of the resulting caustic generator surface (see also the discussion in Section 5.4), potentially many other mappings exist for a specific target that might have interesting properties. For example, the optimal transport map by construction avoids fold-overs, which makes the caustic image rather stable under changes in the spatial configuration, e.g. when translating or rotating the piece with respect to the receiver (see also Figure 5.1). If a more fragile caustic image that exhibits more dynamic behavior is desired, mappings with fold-overs might be advantageous.

5.7 Additions and remarks

Our solution can serve as the basis for a number of future explorations. Extending the caustic design process to handle multiple source surfaces, consider the dynamics of caustics as the spatial configuration changes, or optimizing for multiple caustic images in a single object (as in [BBW+13]), are interesting questions for future research.

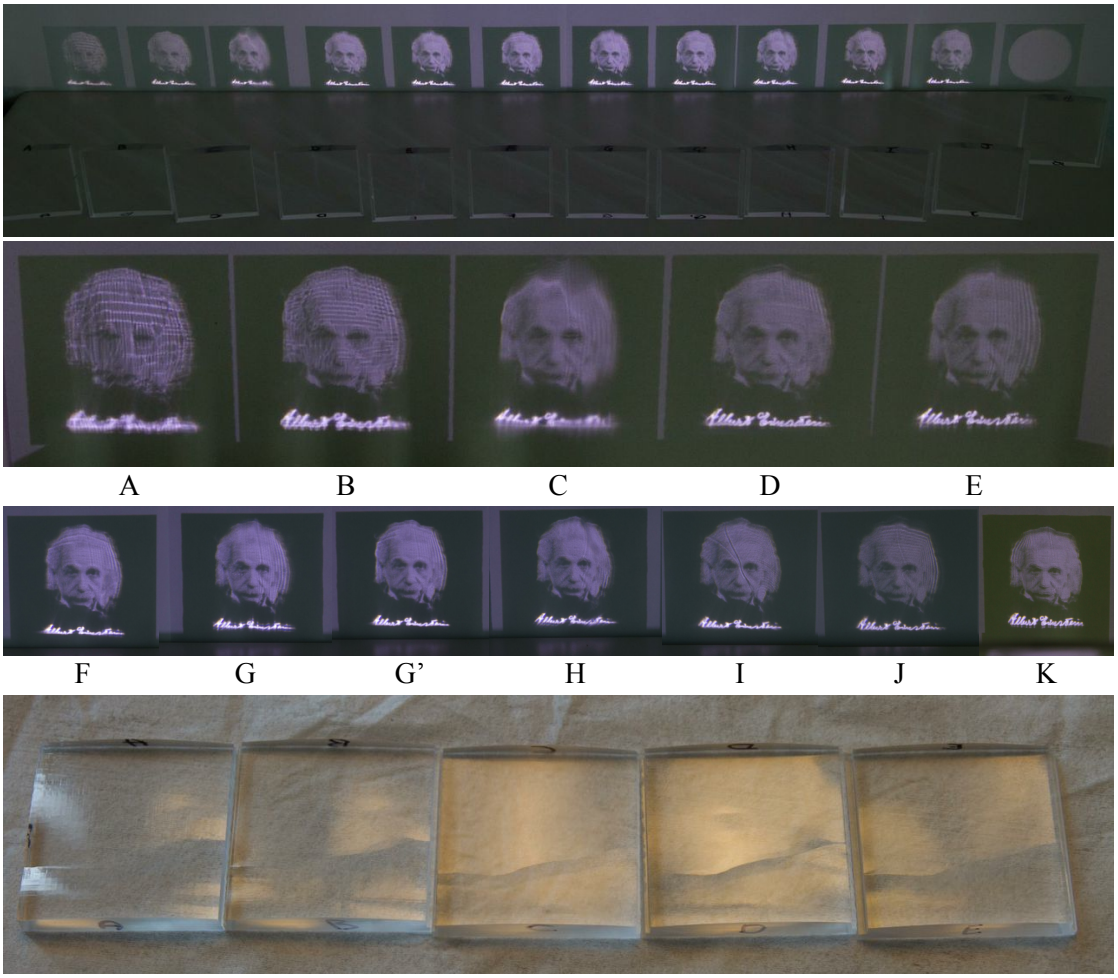


Figure 5.17: Fabrication experiments for the EINSTEIN model. Experiments A-J in order are shown on top with a uniform circle at right. Close-ups of pieces A-K show the jittering artifacts as well as the turning line (middle two rows). Finally, differences on the pieces themselves can be noticed visually at bottom.

Fabrication. The physical prototypes are fabricated in acrylic (PMMA, IOR: 1.49) using the Mikron HSM 400U 5-axis CNC milling machine. Milling is performed in three passes: one with a 10 mm diameter standard drafting mill, and two with a 4 mm diameter diamond drill. The final pass is done with a cusp height of 1.5 microns at 36,000 RPM. After milling, polishing is sometimes necessary, this is done by hand with a PMMA polish paste. We refer to [PEB+13] for more details and experiments related to the fabrication process with glass. Here we detail some experiments with PMMA. Figure 5.17 shows experiments with the EINSTEIN piece, the parameters of each experiment are detailed in Table 5.2. We show only the finishing pass for brevity, and as that is where the most variation occurs. We can see from the experiments that a

common problem with milling is jittering artifacts (seen on the photographs as columns) coming from the vibration of the piece.

		Piece									
Parameter	Units	A	B	C	D	E	F	H	I	J	K
Cutting speed	mm/min	1200	"	"	"	"	"	"	"	"	800
Type of milling		XY	"	"	"	"	YX	"	"	"	XY
Pass size	mm	0.03	"	0.30	0.075	"	"	"	"	"	0.03
Direction (cross)	°	0	0/90	"	"	0	0/90	45/135	45	225	180/90
In-path tolerance	mm	0.02	0.005	0.0001	"	"	"	"	"	"	"
Facet tolerance	*	0.25	"	1	"	"	"	"	"	"	"
Total milling time	min	165	165	30	75	45	75	105	160	45	315

Table 5.2: Fabrication parameters for a subset of experiments A-K for the EINSTEIN model. " signifies that the value is the same to the previous one. All pieces were milled with a spindle speed of 36,000 RPM. For the direction, ω/ϕ signifies that it was milled a first time along an axis $\omega°$ counter-clockwise to the x-axis and a second time along the axis $\phi°$ to the x-axis. Facet tolerance is a multiplicative factor of the in-path tolerance. Experiments G and G' are identical to F except for the emulsion used, and G' was rotated by 180°. I and J use different milling strategies.

Additional results An example of adding color can be seen in Figure 5.10. This is done by applying a printed color film and applying the inverse mapping from caustic to piece to lookup the color for each pixel. We have also experimented with another option of achieving color. We separate a target image into its constituent red, green, and blue color channels. Then we find a generator surface for each channel. Using a red, green, and blue colored light we can then overlay each of the surfaces to produce the target. The result can be seen in Figure 5.18.

Finally, an early experiment of achieving high-contrast caustics was done with the EPFL logo as a target. The result can be seen in Figure 5.19.

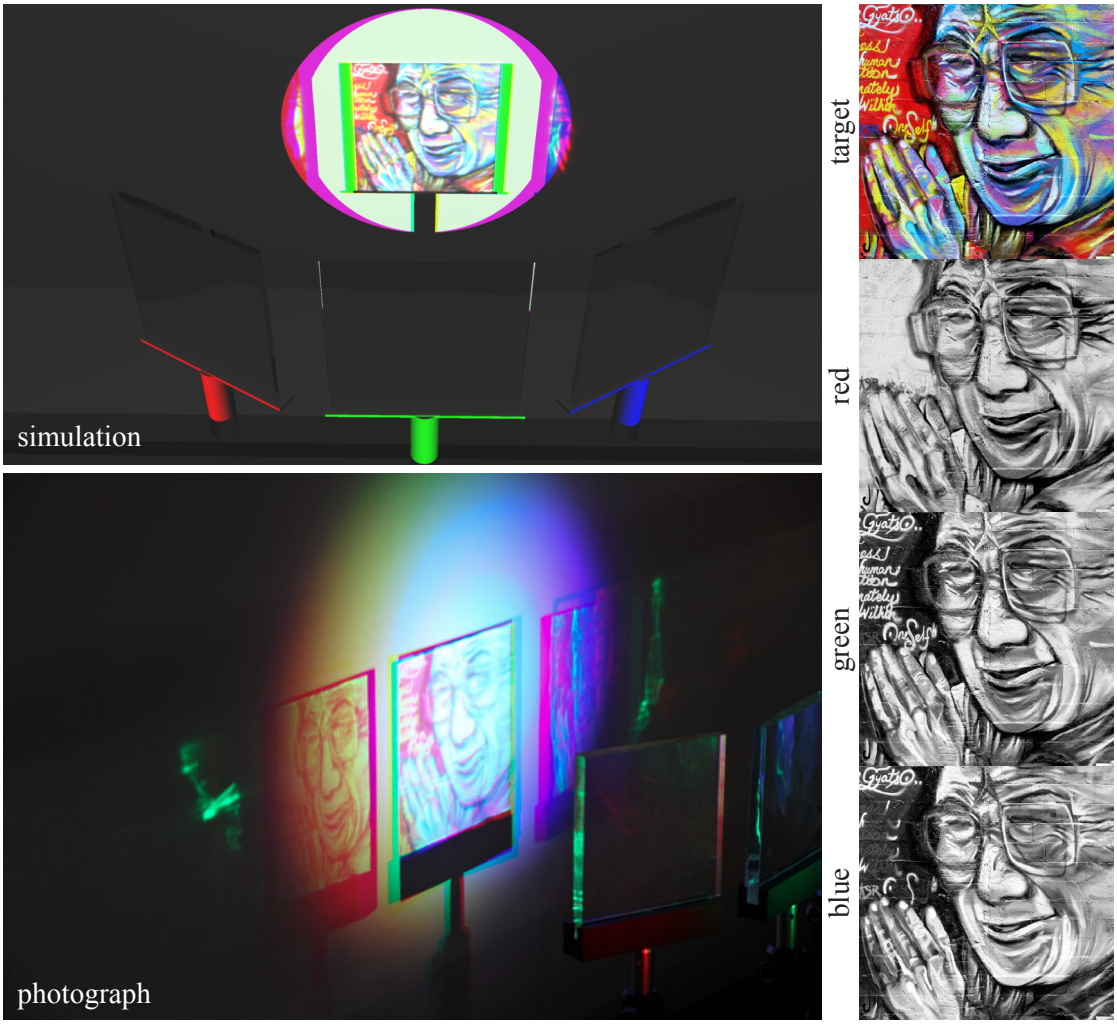


Figure 5.18: A caustic of the graffiti image of the Dalai Lama can be achieved by using separate caustic pieces for each of the red, green, and blue color channels, and colored light sources.

Figure 5.19: A caustic piece showing the EPFL logo illuminated by sunlight.

Chapter 6

Conclusion

This thesis has presented three computational methods for design scenarios with performative constraints. As such, the goal of this thesis is to inspire future work in aiding design through efficient modeling of constraints.

We have shown that interlocking planar pieces offer rich design possibilities for creating compelling 3D models that can be fabricated with low-cost machinery and assembled without glue, screws, or other means of support. We demonstrate that optimization is crucial for exploiting the full creative potential of this type of construction. Only through computational means can we handle the intricate coupling of fabrication and assembly constraints that lead to a highly complex design space. Our approach effectively hides this complexity, allowing even non-expert users to quickly create interesting 3D designs.

We have shown that it is possible to automatically create assembly instructions to build a mesh from LEGO bricks using simple graph-based algorithms. Our method is faster and more accurate than the existing approaches and has the advantage of reporting whether the model is solid or if it has some weaknesses. We can also account for brick type limits and colors, as well as reducing brick count by hollowing.

The first two works deal with the application of graph theory in construction, and this can potentially lead to a rich area of future work. Large-scale construction often involves coordinating many different elements and teams which must be coordinated. Graph-based optimization is particularly useful when one wants to build a larger object out of component building blocks as these naturally form hierarchies and cycles of construction. Possible future work on algorithms on these graphs is reusing building blocks thereby saving time and money by cutting many pieces at once without reprogramming CNC machines.

We have shown how an optimal transport formulation in combination with an iterative 3D optimization provides a powerful new tool for computational caustic design. Our method achieves comparable results to previous methods for low-contrast images, but yields significantly better results for high-contrast caustics. This is achieved by explicit modeling of singularities and automatic placement of discontinuities in the surface normals, made possible by an adaptive discretization that conforms to the target irradiance distribution. As a consequence, our algorithm significantly broadens the kind of caustic images that can be produced, attaining a new visual quality not possible before. Optimal transport is particularly useful to enable smooth mappings with discontinuities. This is directly applicable to equiareal texture maps but might have other applications within lighting or sound such as projectors, displays, or even acoustic mirrors.

Each of these optimization algorithms are examples of performative form-finding: finding a geometric shape/configuration that maximizes a performative goal. These kind of optimizations have many other potential applications, for example in geometry or image editing, digital fabrication, or physical simulation. However, these cases are only the beginning of an exploration into what is possible. The ideas presented open up numerous new research questions for computer graphics and related fields. Additionally, there is still further work to be done in applying these techniques outside of academia. The eventual goal is tools that can be easily and quickly used in design without needing the designer to worry about low-level details. While it might be some time before these are ready, the most valuable applications, and also the best research questions, can only come when artists and architects are actively using such tools in their own projects. To further this goal, it is important to try to combine performative constraints into a framework easily used by architects and designers. Further research will hopefully not only expand the possibilities of fabrication techniques but allow designers to work within a minimal set of constraints in a manner that is general and intuitive.

Bibliography

[AHA98] Franz Aurenhammer, Friedrich Hoffmann, and Boris Aronov. Minkowski-type theorems and least-squares clustering. *Algorithmica*, 20(1):61–76, 1998.

[AMO13] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. https://code.google.com/p/ceres-solver/, 2013.

[APH+03] Maneesh Agrawala, Doantam Phan, Julie Heiser, John Haymaker, Jeff Klingner, Pat Hanrahan, and Barbara Tversky. Designing effective step-by-step assembly instructions. *ACM Transactions on Graphics (TOG)*, 22(3):828–837, July 2003.

[BBJP12] Moritz Bächer, Bernd Bickel, Doug L. James, and Hanspeter Pfister. Fabricating articulated characters from skinned meshes. *ACM Trans. Graph.*, 31(4):47:1–47:9, July 2012.

[BBW+12] Axel Bäuerle, Adrien Bruneton, Rolf Wester, Jochen Stollenwerk, and Peter Loosen. Algorithm for irradiance tailoring using multiple freeform optical surfaces. *Optics express*, 20(13):14477–14485, 2012.

[BBW+13] Adrien Bruneton, Axel Bäuerle, Rolf Wester, Jochen Stollenwerk, and Peter Loosen. High resolution irradiance tailoring using multiple freeform surfaces. *Opt. Express*, 21(9):10563–10571, May 2013.

[BDS+12] Sofien Bouaziz, Mario Deuss, Yuliy Schwartzburg, Thibaut Weise, and Mark Pauly. Shape-up: Shaping discrete geometry with projections. *Computer Graphics Forum*, 31(5):1657–1667, 2012.

[BHT00] Sergey Bobkov, Christian Houdré, and Prasad Tetali. λ_∞, vertex isoperimetry and concentration. *Combinatorica*, 20(2):153–172, 2000.

[BKP+10] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. *Polygon Mesh Processing*. Ak Peters Series. A K Peters, 2010.

[BMD+04] Pablo Benítez, Juan C Min, Oliver Dross, Maikel Herna, Waqidi Falicoff, et al. Simultaneous multiple surface optical design method in three dimensions. *Optical Engineering*, 43(7):1489–1502, 2004.

[BML+14] Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. Projective dynamics: fusing constraint projections for fast simulation. *ACM Transactions on Graphics (TOG)*, 33(4):154, 2014.

[Bog06] V. I. Bogachev. *Measure theory*. Springer, 2006.

[BXM03] S. Boyd, L. Xiao, and A. Mutapcic. Alternating projections. *Lecture notes of EE392o, Stanford University, Autumn Quarter*, 2004, 2003.

[BZK09] David Bommes, Henrik Zimmer, and Leif Kobbelt. Mixed-integer quadrangulation. In *ACM Transactions On Graphics (TOG)*, volume 28, page 77. ACM, 2009.

[CCA+12] Jacques Calì, Dan A. Calian, Cristina Amati, Rebecca Kleinberger, Anthony Steed, Jan Kautz, and Tim Weyrich. 3d-printing of non-assembly, articulated models. *ACM Trans. Graph.*, 31(6):130:1–130:8, November 2012.

[CJL+13] Otis Chodosh, Vishesh Jain, Michael Lindsey, Lyuboslav Panchev, and Yanir A. Rubinstein. On discontinuity of planar optimal transport maps. *arXiv/1312.2929*, 2013.

[CLD+13] Desai Chen, David IW Levin, Piotr Didyk, Pitchaya Sitthi-Amorn, and Wojciech Matusik. Spec2fab: a reducer-tuner model for translating specifications to 3d prints. *ACM Transactions on Graphics (TOG)*, 32(4):135, 2013.

[CLM+13] Duygu Ceylan, Wilmot Li, Niloy J Mitra, Maneesh Agrawala, and Mark Pauly. Designing and fabricating mechanical automata from mocap sequences. *ACM Trans. Graph.*, 32(6):186, 2013.

[CPMS14] Paolo Cignoni, Nico Pietroni, Luigi Malomo, and Roberto Scopigno. Field-aligned mesh joinery. *ACM Trans. on Graphics*, 33(1):art.11, January 2014.

[CSAD04] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. *ACM Trans. Graph.*, 23:905–914, August 2004.

[CSaLM13] Desai Chen, Pitchaya Sitthi-amorn, Justin T Lan, and Wojciech Matusik. Computing and fabricating multiplanar models. In *Computer Graphics Forum*, volume 32, pages 305–315. Wiley Online Library, 2013.

[CTN+13] Stelian Coros, Bernhard Thomaszewski, Gioacchino Noris, Shinjiro Sueda, Moira Forberg, Robert W Sumner, Wojciech Matusik, and Bernd Bickel. Computational design of mechanical characters. *ACM Transactions on Graphics (TOG)*, 32(4):83, 2013.

[DBD+13] Bailin Deng, Sofien Bouaziz, Mario Deuss, Juyong Zhang, Yuliy Schwartzburg, and Mark Pauly. Exploring local modifications for constrained meshes. *Computer Graphics Forum*, 32(2pt1):11–20, 2013.

[DBD+14] Bailin Deng, Sofien Bouaziz, Mario Deuss, Alexandre Kaspar, Yuliy Schwartzburg, and Mark Pauly. Interactive design exploration for constrained meshes. *Computer-Aided Design*, 2014.

[DDSD03] X. Décoret, F. Durand, F.X. Sillion, and J. Dorsey. Billboard clouds for extreme model simplification. *ACM Trans. Graph.*, 22(3):689–696, 2003.

[DGAO+13] Fernando De Goes, Pierre Alliez, Houman Owhadi, Mathieu Desbrun, et al. On the equilibrium of simplicial masonry structures. *ACM Transactions on Graphics (TOG)*, 32(4), 2013.

[dGBOD12] Fernando de Goes, Katherine Breeden, Victor Ostromoukhov, and Mathieu Desbrun. Blue noise through optimal transport. *ACM Transactions on Graphics (TOG)*, 31(6):171, 2012.

[DHL14] Jeremie Dumas, Jean Hergel, and Sylvain Lefebvre. Bridging the gap: Automated steady scaffoldings for 3d printing. *ACM Transactions on Graphics (TOG)*, 33(4), 2014.

[DLS94] Rainer G Dorsch, Adolf W Lohmann, and Stefan Sinzinger. Fresnel ping-pong algorithm for two-plane computer-generated hologram display. *Applied optics*, 33(5):869–875, 1994.

[DO07] E.D. Demaine and J. O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, 2007.

[Dow93] N.E. Dowling. *Mechanical behavior of materials: engineering methods for deformation, fracture, and fatigue*. Prentice Hall Englewood Cliffs, NJ, 1993.

[DPF13] Guido De Philippis and Alessio Figalli. The Monge-Ampère equation and its link to optimal transportation. *arXiv preprint arXiv:1310.6167*, 2013.

[DPW11] Bailin Deng, Helmut Pottmann, and Johannes Wallner. Functional webs for freeform architecture. In *Computer Graphics Forum*, volume 30, pages 1369–1378. Wiley Online Library, 2011.

[DPW+14] Mario Deuss, Daniele Panozzo, Emily Whiting, Yang Liu, Philippe Block, Olga Sorkine-Hornung, and Mark Pauly. Assembling self-supporting structures. *ACM Transactions on Graphics (TOG)*, 2014. to appear.

[EDS+10] Michael Eigensatz, Mario Deuss, Alexander Schiftner, Martin Kilian, Niloy J Mitra, Helmut Pottmann, and Mark Pauly. Case studies in cost-optimized paneling of architectural freeform surfaces. *Advances in Architectural Geometry 2010*, pages 49–72, 2010.

[EKS+10] Michael Eigensatz, Martin Kilian, Alexander Schiftner, Niloy J Mitra, Helmut Pottmann, and Mark Pauly. Paneling architectural freeform surfaces. In *ACM Transactions on Graphics (TOG)*, volume 29, page 45. ACM, 2010.

[EP09] Michael Eigensatz and Mark Pauly. Positional, metric, and curvature control for constraint-based surface deformation. In *Computer Graphics Forum*, volume 28, pages 551–558. Wiley Online Library, 2009.

[FCR09] Florian R Fournier, William J Cassarly, and Jannick P Rolland. Designing freeform reflectors for extended sources. In *SPIE Optical Engineering+ Applications*, pages 742302–742302. International Society for Optics and Photonics, 2009.

[FDL10] Manuel Finckh, Holger Dammertz, and Hendrik PA Lensch. Geometry construction from caustic images. In *Computer Vision–ECCV 2010*, pages 464–477. Springer, 2010.

[FHGJ13] Zexin Feng, Lei Huang, Mali Gong, and Guofan Jin. Beam shaping system design using double freeform optical surfaces. *Optics express*, 21(12):14728–14735, 2013.

[FHJG13] Zexin Feng, Lei Huang, Guofan Jin, and Mali Gong. Designing double freeform optical surfaces for controlling both irradiance and wavefront. *Optics express*, 21(23):28693–28701, 2013.

[Fou10] Florian Fournier. *Freeform reflector design with extended sources*. PhD thesis, University of Central Florida Orlando, Florida, 2010.

[FZW+13] F.Z. Fang, X.D. Zhang, A. Weckenmann, G.X. Zhang, and C. Evans. Manufacturing and measurement of freeform optics. *CIRP Annals - Manufacturing Technology*, 62(2), 2013.

[GHP98] Rebecca A. H. Gower, Agnes Eileen Heydtmann, and Henrik G. Petersen. *LEGO: Automated Model Construction*, pages 81–94. Jens Gravesen and Poul Hjorth, 1998.

[Gla89] Andrew S Glassner. *An introduction to ray tracing*. Morgan Kaufmann, 1989.

[Gla02] A. Glassner. Interactive pop-up card design. 1. *Computer Graphics and Applications, IEEE*, 22(1):79–86, 2002.

[GO03] T. Glimm and V. Oliker. Optical design of single reflector systems and the Monge-Kantorovich mass transfer problem. *Journal of Mathematical Sciences*, 117(3), 2003.

[GSFD+14] Akash Garg, Andrew O Sageman-Furnas, Bailin Deng, Yonghao Yue, Eitan Grinspun, Mark Pauly, and Max Wardetzky. Wire mesh design. *ACM Transactions on Graphics (TOG)*, 33(4):66, 2014.

[GWS04] Johannes Günther, Ingo Wald, and Philipp Slusallek. Realtime caustics using distributed photon mapping. In *Proceedings of the Fifteenth Eurographics conference on Rendering Techniques*, pages 111–121. Eurographics Association, 2004.

[Har07] G. Hart. Modular kirigami. *Journal of Mathematics and the Arts*, 1(1):21–28, 2007.

[HBA12] Krisitian Hildebrand, Bernd Bickel, and Marc Alexa. crdbrd: Shape fabrication by sliding planar pieces. *Comput. Graph. Forum*, 31(2), 2012.

[HGH12] T Heßling, U Geyer, and A Hellwig. Free-form glass reflectors for non-trivial illumination applications with extended sources. 2012.

[HIH+13] Matthias B. Hullin, Ivo Ihrke, Wolfgang Heidrich, Tim Weyrich, Gerwin Damberg, and Martin Fuchs. Computational fabrication and display of material appearance. In *Eurographics State-of-the-Art Report*, 2013.

[II08] Yuki Igarashi and Takeo Igarashi. Pillow: interactive flattening of a 3d model for plush toy design. In *Smart Graphics*, pages 1–7. Springer, 2008.

[IIM12] Yuki Igarashi, Takeo Igarashi, and Jun Mitani. Beady: interactive beadwork design and construction. *ACM Transactions on Graphics (TOG)*, 31(4):49, 2012.

[IIS08] Yuki Igarashi, Takeo Igarashi, and Hiromasa Suzuki. Knitting a 3d model. In *Computer Graphics Forum*, volume 27, pages 1737–1743. Wiley Online Library, 2008.

[Jac07] B. Jackson. Notes on the rigidity of graphs. In *Levico Conference Notes*, 2007.

[Joh10] S.G. Johnson. The nlopt nonlinear-optimization package, 2010.

[KEN+12] Thomas Kiser, Michael Eigensatz, Minh Man Nguyen, Philippe Bompas, and Mark Pauly. Architectural caustics - controlling light with geometry. In *Advances in Architectural Geometry*. Springer, 2012.

[Kil06] Axel Kilian. *Design exploration through bidirectional modeling of constraints*. PhD thesis, Massachusetts Institute of Technology, 2006.

[KP12] Thomas Kiser and Mark Pauly. Caustic art. Technical report, EPFL, 2012.

[Kra94] D. Kraft. Algorithm 733: Tomp–fortran modules for optimal control calculations. *ACM Transactions on Mathematical Software (TOMS)*, 20(3):262–281, 1994.

[LBRM12] Linjie Luo, Ilya Baran, Szymon Rusinkiewicz, and Wojciech Matusik. Chopper: Partitioning models into 3D-printable parts. *ACM Transactions on Graphics*, 2012. To appear.

[LFL09] Kui-Yip Lo, Chi-Wing Fu, and Hongwei Li. 3d polyomino puzzle. *ACM Trans. Graph.*, 28:157:1–157:8, December 2009.

[LGX+13] Anat Levin, Daniel Glasner, Ying Xiong, Frédo Durand, William Freeman, Wojciech Matusik, and Todd Zickler. Fabricating brdfs at high spatial resolution using wave optics. *ACM Transactions on Graphics (TOG)*, 32(4):144, 2013.

[LJGH11] Xian-Ying Li, Tao Ju, Yan Gu, and Shi-Min Hu. A geometric study of v-style pop-ups: Theories and algorithms. *ACM Trans. Graph.*, 30(4):98:1–10, 2011.

[LN89] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[LNLRL13] TV Le-Nguyen, KL Low, Jr C Ruiz, and SN Le. Automatic paper sliceform design from 3d solid models. *IEEE transactions on visualization and computer graphics*, 2013.

[LNR10] Kera Lagios, Jeff Niemasz, and Christoph F Reinhart. Animated building performance simulation (abps)-linking rhinoceros/grasshopper with radiance/daysim. *Proceedings of SimBuild*, 2010.

[LOMI11] Manfred Lau, Akira Ohgawara, Jun Mitani, and Takeo Igarashi. Converting 3d furniture models to fabricatable parts and connectors. *ACM Trans. Graph.*, 30:85:1–85:6, August 2011.

[LPS+13] Yang Liu, Hao Pan, John Snyder, Wenping Wang, and Baining Guo. Computing self-supporting surfaces by regular triangulation. *ACM Transactions on Graphics (TOG)*, 32(4):92, 2013.

[LSH+10] X.Y. Li, C.H. Shen, S.S. Huang, T. Ju, and S.M. Hu. Popup: automatic paper architectures from 3d models. *ACM Trans. Graph.*, 29(4):111, 2010.

[LSS+14] Cheryl Lau, Yuliy Schwartzburg, Appu Shaji, Zahra Sadeghipoor, and Sabine Süsstrunk. Creating personalized jigsaw puzzles. In *Proc. 12th International Symposium on Non-Photorealistic Animation and Rendering*, number EPFL-CONF-199960, 2014.

[LSZ+14] Lin Lu, Andrei Sharf, Haisen Zhao, Yuan Wei, Qingnan Fan, Xuelin Chen, Yann Savoye, Changhe Tu, Daniel Cohen-Or, and Baoquan Chen. Build-to-last: Strength to weight 3d printed objects. *ACM Transactions on Graphics (TOG)*, 33(4), 2014.

[Lé14] Bruno Lévy. A numerical algorithm for *l_2* semi-discrete optimal transport in 3d. *arXiv preprint arXiv:1409.1279*, 2014.

[MA08] Jace Miller and Ergun Akleman. Edge-based intersected polyhedral paper sculptures constructed by interlocking slitted planar pieces. *Proceedings of the 2008 Bridges Conference on Mathematical Connections in Art, Music, and Science*, 2008.

[Mag13] S. Magarill. Fast implementation of oliker's ellipses technology to build free form reflector, 2013.

[Mér11] Quentin Mérigot. A multiscale approach to optimal transport. In *Computer Graphics Forum*, volume 30, pages 1583–1592. Wiley Online Library, 2011.

[MGE07] F. Massarwi, C. Gotsman, and G. Elber. Papercraft models using generalized cylinders. In *Computer Graphics and Applications, 2007. PG'07. 15th Pacific Conference on*, pages 148–157. IEEE, 2007.

[MI07] Yuki Mori and Takeo Igarashi. Plushie: an interactive design system for plush toys. In *ACM Transactions on Graphics (TOG)*, volume 26, page 45. ACM, 2007.

[MKB13] Stefanie Mueller, Bastian Kruck, and Patrick Baudisch. Laserorigami: laser-cutting 3d objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2585–2592. ACM, 2013.

[MLB12] Stefanie Mueller, Pedro Lopes, and Patrick Baudisch. Interactive construction: Interactive fabrication of functional mechanical devices. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 599–606, New York, NY, USA, 2012. ACM.

[MMG+14] Stefanie Mueller, Tobias Mohr, Kerstin Guenther, Johannes Frohnhofen, and Patrick Baudisch. fabrickation: fast 3d printing of functional objects by integrating construction kit building blocks. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 3827–3834. ACM, 2014.

[MP09] Niloy J Mitra and Mark Pauly. Shadow art. In *ACM Transactions on Graphics (TOG)*, volume 28, page 156. ACM, 2009.

[MS04] J. Mitani and H. Suzuki. Making papercraft toys from meshes using strip-based approximate unfolding. *ACM Trans. Graph.*, 23(3):259–263, 2004.

[MS10] J Michael McCarthy and Gim Song Soh. *Geometric design of linkages*, volume 11. Springer, 2010.

[MSM11] J. McCrae, K. Singh, and N.J. Mitra. Slices: A shape-proxy based on planar sections. *ACM Trans. on Graph.(Proc. SIGGRAPH Asia)*, 30(6), 2011.

[MUS14] James McCrae, Nobuyuki Umetani, and Karan Singh. Flatfitfab: Interactive modeling with planar sections, 2014.

[NT03] F.S. Nooruddin and G. Turk. Simplification and repair of polygonal models using volumetric techniques. *Visualization and Computer Graphics, IEEE Transactions on*, 9(2):191–205, 2003.

[Oli13] Vladimir Oliker. Differential equations for design of a freeform single lens with prescribed irradiance properties. *Optical Engineering*, 53(3):031302, 2013.

[Oxm09] Rivka Oxman. Performative design: a performance-based model of digital architectural design. *Environment and Planning B: Planning and Design*, 36(6):1026–1037, 2009.

[PBSH13] Daniele Panozzo, Philippe Block, and Olga Sorkine-Hornung. Designing unreinforced masonry models. *ACM Transactions on Graphics (TOG)*, 32(4):91, 2013.

[PEB+13] Mark Pauly, Michael Eigensatz, Philippe Bompas, Florian Rist, and Raimund Krenmuller. Controlling caustics. *Glass Performance Days*, 2013.

[Pet01] Pavel Petrovic. Solving the LEGO brick layout problem using evolutionary algorithms. Technical report, Norwegian University of Science and Technology, 2001.

[PJJ+11] Marios Papas, Wojciech Jarosz, Wenzel Jakob, Szymon Rusinkiewicz, Wojciech Matusik, and Tim Weyrich. Goal-based caustics. *Computer Graphics Forum*, 30(2):503–511, 2011.

[PKK00] M Pauly, T Kollig, and A Keller. Metropolis light transport for participating media. In *Rendering Techniques 2000. Proceedings of the Eurographics Workshop*, number EPFL-CONF-149336, pages 11–391, 2000.

[PLPZ12] Daniele Panozzo, Yaron Lipman, Enrico Puppo, and Denis Zorin. Fields on symmetric surfaces. *ACM Trans. Graph.*, 31(4):111, 2012.

[POG13] Roi Poranne, Elena Ovreiu, and Craig Gotsman. Interactive planarization and optimization of 3d meshes. In *Computer Graphics Forum*, volume 32, pages 152–163. Wiley Online Library, 2013.

[Pot13] Helmut Pottmann. Architectural geometry and fabrication-aware design. *Nexus Network Journal*, 15(2):195–208, 2013.

[PP05] Gustavo Patow and Xavier Pueyo. A survey of inverse surface design from light transport behavior specification. In *Computer Graphics Forum*, volume 24, pages 773–789. Wiley Online Library, 2005.

[PWLSH13] Romain Prévost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. Make It Stand: Balancing shapes for 3D fabrication. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, 32(4):81:1–81:10, 2013.

[RM02] Harald Ries and Julius Muschaweck. Tailored freeform optical surfaces. *J. Opt. Soc. Am. A*, 19(3):590–595, Mar 2002.

[RW07] J. Rubinstein and G. Wolansky. Intensity control with a free-form lens. *Journal of the Optical Society of America*, 24(2), 2007.

[SCMI13] Daniel Saakes, Thomas Cambazard, Jun Mitani, and Takeo Igarashi. Paccam: Material capture and interactive 2d packing for efficient material usage on cnc cutting machines. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 441–446, New York, NY, USA, 2013. ACM.

[Séq12] C.H. Séquin. Prototyping dissection puzzles with layered manufacturing. *Fabrication and Sculpture Track, Shape Modeling International Conference*, 2012.

[SFG+13] Peng Song, Chi-Wing Fu, Prashant Goswami, Jianmin Zheng, Niloy J Mitra, and Daniel Cohen-Or. Reciprocal frame structures made easy. *ACM Transactions on Graphics (TOG)*, 32(4):94, 2013.

[Sib81] Robin Sibson. A brief description of natural neighbour interpolation. *Interpreting multivariate data*, 21, 1981.

[SLMI11] G. Saul, M. Lau, J. Mitani, and T. Igarashi. Sketchchair: An all-in-one chair design system for end users. In *Proceedings of the fifth international conference on tangible, embedded, and embodied interaction*, pages 73–80. ACM, 2011.

[SNR11] Jon Sargent, Jeffrey Niemasz, and Christoph F Reinhart. Shaderade: Combining rhinoceros and energyplus for the design of static exterior shading devices. In *Building Simulation*, pages 1–9, 2011.

[SP12] Yuliy Schwartzburg and Mark Pauly. Design and optimization of orthogonally intersecting planar surfaces. In *Computational Design Modelling*, pages 191–199. Springer, 2012.

[SP13] Yuliy Schwartzburg and Mark Pauly. Fabrication-aware design with intersecting planar pieces. *Computer Graphics Forum (Proc. of Eurographics 2013)*, 32(2pt3):317–326, 2013.

[SPC09] L.F.M.S. Silva, V.F. Pamplona, and J.L.D. Comba. Legolizer: A real-time system for modeling and rendering LEGO representations of boundary models. In *Computer Graphics and Image Processing (SIBGRAPI)*, 2009.

[SSLP14] A Schulz, A Shamir, DIW Levin, and Sitthi-Amorn, P. Design and fabrication by example. 2014.

[SSSD+14] Stefan Schneegass, Alireza Sahami Shirazi, Tanja Döring, David Schmid, and Albrecht Schmidt. Natcut: an interactive tangible editor for physical object fabrication. In *CHI'14 Extended Abstracts on Human Factors in Computing Systems*, pages 1441–1446. ACM, 2014.

[STK+14] Mélina Skouras, Bernhard Thomaszewski, Peter Kaufmann, Akash Garg, Bernd Bickel, Eitan Grinspun, and Markus Gross. Designing inflatable structures. *ACM Transactions on Graphics (TOG)*, 33(4):63, 2014.

[STL06] I. Shatz, A. Tal, and G. Leifman. Paper craft models from meshes. *The Visual Computer*, 22(9):825–834, 2006.

[STTP14] Yuliy Schwartzburg, Romain Testuz, Andrea Tagliasacchi, and Mark Pauly. High-contrast computational caustic design. *ACM Trans. Graph. (Proc. of SIGGRAPH 2014)*, 33(4):74:1–74:11, July 2014.

[SVB+12] Ondrej Stava, Juraj Vanek, Bedrich Benes, Nathan Carr, and Radomír Měch. Stress relief: improving structural strength of 3d printable objects. *ACM Transactions on Graphics (TOG)*, 31(4):48, 2012.

[SY13] Xing Shi and Wenjie Yang. Performance-driven architectural design and optimization technique from a perspective of architects. *Automation in Construction*, 32:125–135, 2013.

[Tac10] Tomohiro Tachi. Freeform rigid-foldable structure using bidirectionally flat-foldable planar quadrilateral mesh. In C. Ceccato et al., editors, *Advances in Architectural Geometry 2010*, pages 87–102. Springer, 2010.

[TCG+14] Bernhard Thomaszewski, Stelian Coros, Damien Gauge, Vittorio Megaro, Eitan Grinspun, and Markus Gross. Computational design

of linkage-based characters. *ACM Transactions on Graphics (TOG)*, 33(4):64, 2014.

[TSG+14] Chengcheng Tang, Xiang Sun, Alexandra Gomes, Johannes Wallner, and Helmut Pottmann. Form-finding with polyhedral meshes made simple. *ACM Transactions on Graphics (TOG)*, 33(4):70, 2014.

[TSP13] Romain Testuz, Yuliy Schwartzburg, and Mark Pauly. Automatic generation of constructable brick sculptures. In *Eurographics 2013-Short Papers*, pages 81–84. The Eurographics Association, 2013.

[UIM12] Nobuyuki Umetani, Takeo Igarashi, and Niloy J. Mitra. Guided exploration of physically valid shapes for furniture design. *ACM Transactions on Graphics*, 31(4), 2012.

[Vax12] Amir Vaxman. Modeling polyhedral meshes with affine maps. In *Computer Graphics Forum*, volume 31, pages 1647–1656. Wiley Online Library, 2012.

[VHWP12] Etienne Vouga, Mathias Höbinger, Johannes Wallner, and Helmut Pottmann. Design of self-supporting surfaces. *ACM Transactions on Graphics (TOG)*, 31(4):87, 2012.

[Vil09] C. Villani. *Optimal transport: old and new*. Springer Verlag, 2009.

[VWRKM13] Kiril Vidimče, Szu-Po Wang, Jonathan Ragan-Kelley, and Wojciech Matusik. Openfab: A programmable pipeline for multi-material fabrication. *ACM Transactions on Graphics (TOG)*, 32(4):136, 2013.

[vZS08] L. van Zijl and E. Smal. Cellular automata with cell clustering. *Automata-2008: Theory and Applications of Cellular Automata*, page 425, 2008.

[Win05] D.V. Winkler. Automated brick layout. *BrickFest*, 2005.

[WLMI10] Karl D.D. Willis, Juncong Lin, Jun Mitani, and Takeo Igarashi. Spatial sketch: bridging between movement & fabrication. In *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction*, pages 5–12. ACM, 2010.

[WLY+08] Wenping Wang, Yang Liu, Dongming Yan, Bin Chan, Ruotian Ling, and Feng Sun. Hexagonal meshes with planar faces. *Dept. of CS, HKU, Tech. Rep*, 2008.

[WPMR09] Tim Weyrich, Pieter Peers, Wojciech Matusik, and Szymon Rusinkiewicz. Fabricating microgeometry for custom surface reflectance. *ACM Transactions on Graphics (TOG)*, 28(3):32, 2009.

[WQL07] Lin Wang, Keyuan Qian, and Yi Luo. Discontinuous free-form lens design for prescribed irradiance. *Applied optics*, 46(18):3716–3723, 2007.

[WS03] Michael Wand and Wolfgang Straßer. Real-time caustics. In *Computer Graphics Forum*, volume 22, pages 611–620. Wiley Online Library, 2003.

[WSW+12] Emily Whiting, Hijung Shin, Robert Wang, John Ochsendorf, and Frédo Durand. Structural optimization of 3d masonry buildings. *ACM Transactions on Graphics (TOG)*, 31(6):159, 2012.

[WWY+13] Weiming Wang, Tuanfeng Y Wang, Zhouwang Yang, Ligang Liu, Xin Tong, Weihua Tong, Jiansong Deng, Falai Chen, and Xiuping Liu. Cost-effective printing of 3d objects with skin-frame structures. *ACM Transactions on Graphics (TOG)*, 32(6):177, 2013.

[XLF+11] S. Xin, C.F. Lai, C.W. Fu, T.T. Wong, Y. He, and D. Cohen-Or. Making burr puzzles from 3d models. *ACM Trans. Graph.*, 30(4):97, 2011.

[XZWC14] Wuyuan Xie, Yunbo Zhang, Charlie C. L. Wang, and Ronald C.-K. Chung. Surface-from-gradients: An approach based on discrete geometry processing. June 2014.

[YIC+12] Yonghao Yue, Kei Iwasaki, Bing-Yu Chen, Yoshinori Dobashi, and Tomoyuki Nishita. Pixel art with refracted light by rearrangeable sticks. *Comp. Graph. Forum*, 31(2pt3):575–582, May 2012.

[YIC+14] Yonghao Yue, Kei Iwasaki, Bing-Yu Chen, Yoshinori Dobashi, and Tomoyuki Nishita. Poisson-based continuous surface generation for goal-based caustics. *ACM Trans. Graph.*, 33(3):31:1–31:7, May 2014.

[Yvi13] Mariette Yvinec. 2D triangulations. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.3 edition, 2013.

[YYPM11] Yong-Liang Yang, Yi-Jun Yang, Helmut Pottmann, and Niloy J Mitra. Shape space exploration of constrained meshes. *ACM Trans. Graph.*, 30(6):124, 2011.

[ZLAK14] Henrik Zimmer, Florent Lafarge, Pierre Alliez, and Leif Kobbelt. Zometool shape approximation. *Graphical Models*, 2014.

[ZSW10] Mirko Zadravec, Alexander Schiftner, and Johannes Wallner. Designing quad-dominant meshes with planar faces. In *Computer Graphics Forum*, volume 29, pages 1671–1679. Wiley Online Library, 2010.

[ZXS+12] Lifeng Zhu, Weiwei Xu, John Snyder, Yang Liu, Guoping Wang, and Baining Guo. Motion-guided mechanical toy modeling. *ACM Trans. Graph.*, 31(6):127:1–127:10, November 2012.

[ZZ12] Z. Zhang and X. Zheng. The representation of line Dirac delta function along a space curve. *arXiv preprint arXiv:1209.3221*, September 2012.

Yuliy Schwartzburg

Rue de Genève 66B
1004 Lausanne VD, Switzerland

yuliy.schwartzburg@epfl.ch
www.chateaunoir.net

Profile

Computer graphics researcher interested in digital fabrication and architectural geometry currently working on computational caustic design.

Education

École Polytechnique Fédérale de Lausanne; Switzerland – Candidate Docteur ès Sciences, EDIC Computer and Communication Sciences, Computer Graphics and Geometry Lab – 2009-

The Cooper Union for the Advancement of Science and Art; New York, NY – Bachelor of Engineering, Electrical Engineering, Computer Engineering Track – 2004-2008
Cumulative GPA 3.8/4.0

Experience

Ph.D. Student, Ecole polytechnique fédérale de Lausanne Lausanne, Switzerland – 2009–Present
Research on architectural geometry and digital fabrication. Published peer-reviewed papers on topics including geometry processing, deformation, complex assembly using laser cut pieces, puzzle generation, and high-contrast computational caustics.

Fulbright Fellow, University of Tsukuba, Graduate School of Systems and Information Engineering, Department of Intelligent Interaction Technologies, Yamashita Laboratory Tsukuba, Japan – 2008–2009
As part of a Fulbright Fellowship to Japan, did research in Human Computer Interaction. Built a public display informed uniquely by Japanese culture that interacts with viewers based on their attention to the display.

Financial Software Developer Intern, Bloomberg L.P. New York, NY – 2008
Worked with the Multicast Library group. Wrote complete multicast file transfer program in C++, implementing a unique application-level protocol, and including many features such as resiliency to high data loss and low memory environments, push-based file sending, daemon functionality, and ability to send directory trees.

Invited Researcher, Osaka University Graduate School of Engineering, Department of Adaptive Machine Systems, Hosoda Laboratory Osaka, Japan – 2006
Performed research in humanoid robotics. Experimented on the McKibben pneumatic actuator and on implementing a resistive sensor for the actuator. Involved programming for Linux and the Renesas H8 Microprocessor. Worked with graduate students and

professors in robotics research, and presented work to assist with the laboratory goal of a humanlike walking robot.

Senior Operator, Cooper Union Computer Center New York, NY – 2005–2008
Managed and supervised student operators and dealt with administrative tasks. Provided technical support to students and faculty; maintained and fixed computers; installed and upgraded software; worked closely with senior IT staff.

Publications

High-contrast Computational Caustic Design
Yuliy Schwartzburg, Romain Testuz, Andrea Tagliasacchi, Mark Pauly
ACM Transactions on Graphics (Proc. of ACM SIGGRAPH 2014)

Creating Personalized Jigsaw Puzzles
Cheryl Lau, *Yuliy Schwartzburg*, Appu Shaji, Zahra Sadeghipoor, Sabine Süsstrunk
International Symposium on Non-Photorealistic Animation and Rendering, 2014

Interactive Design Exploration for Constrained Meshes
Bailin Deng, Sofien Bouaziz, Mario Deuss, Alexandre Kaspar, *Yuliy Schwartzburg*, Mark Pauly
Computer-Aided Design 2014

Fabrication-aware Design with Intersecting Planar Pieces
Yuliy Schwartzburg, Mark Pauly
Computer Graphics Forum (Proc. Eurographics) 2013

Automatic Generation of Constructable Brick Sculptures
Romain Testuz, *Yuliy Schwartzburg*, Mark Pauly
Eurographics 2013 Short Paper

Exploring Local Modifications for Constrained Meshes
Bailin Deng, Sofien Bouaziz, Mario Deuss, Juyong Zhang, *Yuliy Schwartzburg*, Mark Pauly
Computer Graphics Forum (Proc. Eurographics) 2013

Shape-Up: Shaping Discrete Geometry with Projections
Sofien Bouaziz, Mario Deuss, *Yuliy Schwartzburg*, Thibaut Weise, Mark Pauly
Symposium on Geometry Processing 2012

Design and Optimization of Orthogonally Intersecting Planar Surfaces
Yuliy Schwartzburg, Mark Pauly
Computational Design Modelling 2012 (Proc. of the Design Modelling Symposium 2011)

Skills

Computer Languages/Tools: C, C++, MATLAB, UNIX, Linux, Lua, Java, JavaScript, AutoCAD, Maya, LaTeX, Adobe Photoshop, Illustrator, Premiere, Flash
Languages: Fluent in English, French, Russian; Knowledge of Hebrew, Japanese

Honors	Fulbright Fellowship to Japan, 2008-2009; Dean's List: Fall 2004-Spring 2008; Full Tuition Cooper Union Fellowship; Dale E. Zand Prize Society of American Military Engineers Fellowship; Harry and Peggy Ploss Fellowship Schering-Plough, Inc. National Merit Special Scholarship
Memberships	ACM SIGGRAPH, Active Student Member Eta Kappa Nu, Delta Chi Chapter, Former Vice-President Tau Beta Pi, New York Iota Chapter National Society of Collegiate Scholars, Cooper Union Chapter Institute of Electrical and Electronics Engineers